FrodoKEM: A CCA-Secure Learning With Errors Key Encapsulation Mechanism

Lewis Glabush[1], Patrick Longa[2], Michael Naehrig[2], Chris Peikert[3], Douglas Stebila[a,4] and Fernando Virdia[b,5]

[1] EPFL, Lausanne, Switzerland
[2] Microsoft Research, Redmond, United States
[3] University of Michigan, Ann Arbor, United States
[4] University of Waterloo, Waterloo, Canada
[5] King's College London, London, United Kingdom

Abstract. Large-scale quantum computers capable of implementing Shor's algorithm pose a significant threat to the security of the most widely used public-key cryptographic schemes. This risk has motivated substantial efforts by standards bodies and government agencies to identify and standardize quantum-safe cryptographic systems. Among the proposed solutions, lattice-based cryptography has emerged as the foundation for some of the most promising protocols.

This paper describes FrodoKEM, a family of conservative key-encapsulation mechanisms (KEMs) whose security is based on generic, "unstructured" lattices. FrodoKEM is proposed as an alternative to the more efficient lattice schemes that utilize *algebraically structured* lattices, such as the recently standardized ML-KEM scheme. By relying on generic lattices, FrodoKEM minimizes the potential for future attacks that exploit algebraic structures while enabling simple and compact implementations. Our plain C implementations demonstrate that, despite its conservative design and parameterization, FrodoKEM remains practical. For instance, the full protocol at NIST security level 1 runs in approximately 0.97 ms on a server-class processor, and 4.98 ms on a smartphone-class processor.

FrodoKEM obtains (single-target) IND-CCA security using a variant of the Fujisaki–Okamoto transform, applied to an underlying public-key encryption scheme called FrodoPKE. In addition, using a new tool called the Salted Fujisaki–Okamoto (SFO) transform, FrodoKEM is also shown to *tightly* achieve *multi-target* security, without increasing the FrodoPKE message length and with a negligible performance impact, based on the multi-target IND-CPA security of FrodoPKE.

Keywords: Post-Quantum Cryptography · Lattice Cryptography · Key Exchange · Multi-target security · Implementation

1 Introduction

Quantum computing research has had significant implications for cryptography [YN05, KBF+15]. Currently, the most widely used asymmetric (i.e., public-key) cryptographic protocols rely on the conjectured intractability of number-theoretic problems like integer factorization and computing discrete logarithms. However, these problems are known to

E-mail: lewis.glabush@epfl.ch (Lewis Glabush), plonga@microsoft.com (Patrick Longa), mnaehrig@microsoft.com (Michael Naehrig), cpeikert@umich.edu (Chris Peikert), dstebila@uwaterloo.ca (Douglas Stebila), fernando.virdia@kcl.ac.uk (Fernando Virdia)

[a]D.S. was supported by NSERC grants RGPIN-2022-03187 and ALLRP 578463-22.
[b]F.V. was supported by UKRI grant EP/Y02432X/1.

Date of this document: 2025-10-07.

be easy for large-scale quantum computers, so these computers (if they are ever built) would be able to completely break the world's most prevalent cryptography.

Motivated by this potentially catastrophic threat, standards bodies and government agencies have initiated efforts to standardize quantum-safe, or "post-quantum," cryptography—i.e., systems that can be run on today's ordinary computers and networks, and are believed to be secure against quantum attacks. In 2017, the National Institute of Standards and Technology (NIST) launched a large-scale project to select and standardize quantum-safe algorithms for digital signature, encryption, and key-establishment protocols [Nat17]. Among the candidates, schemes based on lattice problems—particularly the learning with errors (LWE) [Reg09] and short integer solution (SIS) [Ajt96] problems, and their variants—emerged as especially promising. A significant milestone was achieved in 2022 when NIST selected two lattice-based schemes for standardization: CRYSTALS-Kyber, a key-encapsulation mechanism renamed as ML-KEM [Nat24b], and CRYSTALS-Dilithium, a signature scheme renamed as ML-DSA [Nat24a]. Both of these schemes are based on the Module-LWE problem, a variant of LWE with additional algebraic structure for efficiency purposes. Additionally, NIST selected Falcon, another lattice-based signature scheme, and SPHINCS$^+$, a hash-based signature scheme.

Despite the promising security and efficiency profiles of the algorithms selected by NIST, several government agencies have expressed a desire for more conservative options with less underlying algebraic structure. The two most notable examples are Classic McEliece [ABC+24], from the code-based family, and FrodoKEM, from the lattice-based family. FrodoKEM made it to Round 3 (as an alternate candidate) of the NIST PQC standardization project, and Classic McEliece to Round 4. Both algorithms have been recommended as conservative alternatives by the German BSI [Fed24], the French ANSSI [Nat23], and the Dutch NLNCSA and AIVD [Gen22].[1] Notably, Classic McEliece and FrodoKEM, alongside ML-KEM, are currently undergoing standardization by the International Organization for Standardization (ISO) [Int24].

In this paper, we describe FrodoKEM, a family of IND-CCA secure key-encapsulation mechanisms (KEMs). FrodoKEM is designed as a conservative yet practical post-quantum construction whose security derives from cautious parameterizations of the well-studied learning with errors (LWE) problem. In turn, LWE has close connections to conjectured-hard problems on generic, algebraically unstructured lattices. This paper presents FrodoKEM as of 2025, incorporating design and analysis updates.

1.1 Pedigree

The core of FrodoKEM is a public-key encryption scheme called FrodoPKE, whose IND-CPA security is tightly related to the hardness of a corresponding learning with errors problem. Here we briefly recall the scientific lineage of these systems. See the surveys [Mic10, Reg10, Pei16a] for further details.

The seminal works of Ajtai [Ajt96] and Ajtai–Dwork [AD97] gave the first cryptographic constructions whose security properties followed from the conjectured *worst-case* hardness of various problems on point *lattices* in $\mathbb{R}^n$. In subsequent years, these works were substantially refined and improved, e.g., in [GGH96, CN97, Mic02, Reg04, MR07]. Notably, in work published in 2005, Regev [Reg09] defined the *learning with errors* (LWE) problem, proved the hardness of (certain parameterizations of) LWE assuming the hardness of various worst-case lattice problems for *quantum* algorithms, and defined a public-key encryption scheme whose IND-CPA security is tightly related to the hardness of LWE.[2]

[1] In the latest edition of the PQC migration handbook [Gen24], the Dutch AIVD, CWI and TNO describe FrodoKEM and Classic McEliece as more conservative options and "strongly support ongoing initiatives aiming to standardise them". The document classifies both algorithms as "acceptable" until their standardization is completed.

[2] As pointed out in [Pei09b], Regev's encryption scheme implicitly contains an (unauthenticated) "approximate" key-exchange protocol analogous to the classic Diffie–Hellman protocol [DH76].

Regev's initial work on LWE was followed by much more, which, among other things:

- provided additional theoretical support for the hardness of various LWE parameterizations (e.g., [Pei09a, ACPS09, BLP+13, DM13, MP13, PRS17]),
- extensively analyzed the concrete security of LWE and closely related lattice problems (e.g., [MR09, CN11, LN13, AFG14, Che13, ACFP15, AFFP14, Laa15a, KF15, APS15, ADPS16, BCD+16, Alb17, AGVW17], among countless others), and
- constructed LWE-based cryptosystems with improved efficiency or additional functionality (e.g., [PW08, PVW08, GPV08, CHKP12, BV11, GSW13, BGG+14, GVW15]).

In particular, in work published in 2011, Lindner and Peikert [LP11] gave a more efficient LWE-based public-key encryption scheme that uses a square public matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$ instead of an oblong rectangular one.

Cryptographic schemes named "Frodo". Since there are now several cryptographic schemes incorporating the name "Frodo", we take a moment to clarify how they are related.

- **"Frodo" / FrodoCCS.** An LWE-based key exchange protocol called "Frodo" was published at the 2016 ACM CCS conference [BCD+16], based on the Lindner–Peikert scheme [LP11] with some modifications, such as: pseudorandom generation of the public matrix $\mathbf{A}$ from a small seed, more balanced key and ciphertext sizes, and new LWE parameters. For clarity, we refer to this as FrodoCCS.
- FrodoKEM **(NIST submission versions).** An LWE-based KEM called FrodoKEM was submitted to the NIST Post-Quantum Cryptography standardization project [NAB+17, NAB+19, NAB+20]. At its heart is a public-key encryption scheme called FrodoPKE, to which a Fujisaki–Okamoto-type transform [FO99a] is applied to obtain a KEM with IND-CCA security. Some differences between FrodoCCS and FrodoKEM/FrodoPKE (NIST submission versions) include:
  - FrodoCCS was described as an unauthenticated key-exchange protocol, which can equivalently be viewed as an IND-CPA-secure KEM, whereas FrodoKEM is designed to be an IND-CCA-secure KEM.
  - FrodoCCS used a "reconciliation mechanism" to extract shared-key bits from approximately equal values (similarly to [DXL12, Pei14, BCNS15, ADPS16]), whereas FrodoKEM used simpler key transport via public-key encryption (as in [Reg09, LP11]).
  - FrodoKEM used significantly "wider" LWE error distributions than FrodoCCS, which conform to certain worst-case hardness theorems (see Section C.1.5).
  - FrodoKEM used different symmetric-key primitives than FrodoCCS.

  Note that FrodoPKE and FrodoKEM changed between the first and second rounds of the NIST PQC standardization project.
- FrodoKEM **and** eFrodoKEM **(ISO submission version [ABD+25], and this document).** In response to concerns about multi-ciphertext attacks (see Section 1.3 below), a new version of FrodoKEM was defined that included a *salt* as a countermeasure. As of the ISO submission version, the name eFrodoKEM (with the "e" meaning "ephemeral") refers to the un-salted version (which is equivalent to the NIST Round 3 version of FrodoKEM), and the name FrodoKEM refers to the salted version. The FrodoPKE scheme remains unchanged since the NIST Round 2 version.
- We also use the name "FrodoKEM" to refer to the overall family of schemes consisting of the ephemeral KEM eFrodoKEM and the salted KEM FrodoKEM.

1.2 Structured versus unstructured lattices

Problems that underlie the security of cryptographic protocols often have special structured instances, whose use may offer better efficiency or other advantages, but may also introduce security vulnerabilities that are not present in the general case. The presence and degree of security loss from using structured instances often remains unknown or not well understood.

In lattice-based cryptography, the (plain) LWE problem relates to solving a noisy linear system (modulo a known integer); it can also be interpreted as the problem of decoding a random unstructured lattice from a certain broad class. There are also several variants of LWE where the linear system (and its associated lattice) has additional algebraic structure, which offers advantages in terms of computational efficiency and size. These variants include Ring-LWE [LPR13], Module-LWE [BGV12, LS15] and the NTRU problem [HPS98].

After a good deal of investigation, the state of the art for recommended parameterizations of algebraic LWE variants does not indicate any particular weaknesses in comparison to plain LWE. However, at present there appear to be some gaps between the (quantum) complexity of some *related*, seemingly weaker problems on certain kinds of algebraic lattices and their counterparts on general lattices. (See the next paragraph for details.) Of course, this only represents our current understanding of these problems, which could potentially change with further cryptanalytic effort.

For recommended parameterizations of Ring- and Module-LWE, the current best attacks perform essentially the same as those for plain LWE, apart from some obvious linear-factor (in the ring dimension) savings in time and memory; the same goes for the underlying worst-case problems on ideal and module lattices, versus generic lattices [CN11, Sch13, IKMT14, BNvdP17, Laa15b].[3] However, some conventional NTRU parameterizations admit specialized attacks with significantly better asymptotic performance than on generic lattices with the same parameters [KF15, KF17]. In addition, a series of works [CGS14, CDPR16, CDW17] has yielded a quantum polynomial-time algorithm for very large but *subexponential* $2^{\tilde{O}(\sqrt{n})}$ approximations to the worst-case Shortest Vector Problem on *ideal* lattices over a widely used class of rings (in contrast to just slightly subexponential $2^{O(n \log\log n/\log n)}$ factors obtainable for general lattices [LLL82, Sch87]). Similar results were later obtained for arbitrary number fields, assuming some field-specific exponential-time preprocessing [PHS19]. Note that these subexponential approximation factors are still much larger than the small *polynomial* factors that are typically used in cryptography, and the algorithms from [CGS14, CDPR16, CDW17, PHS19] do not yet have any impact on Ring- or Module-LWE themselves.

1.3 Multi-target security

Multi-key security. Multi-key (also known as multi-user) attacks aim to break security against *any one* of many available public keys.[4] In its call for proposals for the post-quantum standardization process [Nat17], NIST lists "resistance to multi-key attacks" as a "desirable property."

FrodoKEM's primary security target of IND-CCA considers only a single public key, so multi-key attacks fall outside its immediate scope. However, multi-key security generically follows from IND-CCA security by a routine hybrid argument, with linear concrete security loss in the number of keys. In addition, all versions of FrodoKEM (and FrodoCCS before it) include a specific countermeasure against multi-key attacks, namely, a distinct LWE matrix $\mathbf{A}$ for each public key.

Multi-ciphertext security. Multi-ciphertext attacks target a single public key, but aim to break *any one* of many ciphertexts produced under that key. As above, multi-ciphertext security falls outside the scope of FrodoKEM's primary security target of IND-CCA. In contrast to multi-key security, NIST's call for proposals did not mention multi-ciphertext

[3] Some unconventional parameterizations of Ring-LWE were specifically devised to be breakable by certain algebraic attacks [ELOS15, CLS17, CIV16, CLS16]. However, it was later shown that their error distributions are insufficiently "wide" relative to the ring, so they reveal errorless (or nearly so) linear equations and can therefore be broken even more efficiently using elementary, non-algebraic means [CIV16, Pei16b].

[4] In the literature, multi-key attacks are sometimes also known as multi-*target* attacks, but the latter term can also refer to multi-*ciphertext* attacks. So, to avoid ambiguity, we eschew the term "multi-target."

security. However, it can be a desirable feature in applications where a large number of ciphertexts will be encrypted under the same public key.

In 2021, a multi-ciphertext attack against the Round-3 FrodoKEM-640 parameters (for NIST security level 1) and a proposed fix was identified by NIST [Per21]; the same attack (but no fix) was also identified in [Ber22]. The attack exploits FrodoKEM-640's 128-bit message length, and we additionally observed that a similar attack can exploit the 128-bit secret random $\mathsf{seed}_{\mathbf{SE}}$ value (see below for details). Although these attacks do not invalidate any of the security claims from the FrodoKEM submission, they may be of concern for applications with long-lived public keys. For this reason, subsequent versions of FrodoKEM specifically aim for multi-ciphertext security (and multi-target security more generically; see below) via a *salted* version of the Fujisaki–Okamoto transform, and by enlarging $\mathsf{seed}_{\mathbf{SE}}$. For compatibility purposes, and for applications where the number of ciphertexts produced under any single public key is fairly small, there is *ephemeral* FrodoKEM, which is identical to Round-3 FrodoKEM.

The multi-ciphertext attack of [Per21, Ber22] stems from FrodoKEM-640's message length of 128 bits, and the fact that encryption is deterministic (which holds for any KEM built from the Fujisaki–Okamoto transform). When there are n_{c} challenge ciphertexts available, an adversary can sample N distinct messages, and calculate their corresponding ciphertexts and KEM keys. If there is a collision between a challenge ciphertext and a generated ciphertext, the adversary breaks one-wayness, and thereby indistinguishability. The probability of a collision is approximately $n_{\mathsf{c}}N/|M|$, where M is the message space and $n_{\mathsf{c}}, N \ll |M|$. So, with FrodoKEM-640's message space of size $|M| = 2^{128}$, and (say) $n_{\mathsf{c}} = 2^{40}$ challenge ciphertexts, an adversary can break multi-ciphertext security by doing about $N = 2^{88}$ encryptions. A similar attack and analysis also applies to the secret 128-bit $\mathsf{seed}_{\mathbf{SE}}$ value that is used to generate LWE secrets and errors in ciphertexts.

The fix for $\mathsf{seed}_{\mathbf{SE}}$ is trivial: simply increase its length to make collisions infeasible. Similarly, one could use a larger message space, but this cannot be done in FrodoPKE without substantially changing the LWE parameters. Instead, the new version of FrodoKEM adds a *public* salt to make it infeasible for the adversary-generated ciphertexts to collide with challenge ciphertexts.

Multi-target security. Multi-target security unifies both multi-key and multi-ciphertext security. Here, the adversary is given potentially several public keys and ciphertexts, where each ciphertext is generated under one of the public keys of the adversary's choice.[5] To obtain multi-target security, FrodoKEM uses the above-mentioned salted Fujisaki–Okamoto (SFO) transform, and hashing of public keys. The efficacy of these techniques was proven in [GHS25, Gla24].

1.4 Our contributions

In this work, we present and analyze FrodoKEM, a family of key-encapsulation mechanisms that rely on the learning with errors problem to obtain security against known quantum threats. Our focus is on members of the FrodoKEM family presented in the ISO submission [ABD+25], namely: eFrodoKEM, the IND-CCA-secure KEM built from FrodoPKE using the Fujisaki–Okamoto ($\mathsf{FO}^{\not\perp\prime}$) transform; and FrodoKEM, the multi-target-IND-CCA-secure KEM built from FrodoPKE using the salted FO transform. Supporting algorithms for FrodoKEM are presented in Section 3, FrodoPKE is presented in Section 4, and FrodoKEM is presented in Section 5.

Given the existing work on past versions of Frodo in [BCD+16, NAB+17, NAB+19, NAB+20, ABD+25], the specific contributions of this work are as follows.

[5] For a finer-grained notion, one can limit the total number of ciphertexts that may be generated under any single key.

- We apply the analysis of the salted FO transform of [GHS25] to derive results on the multi-target (i.e., multi-key and multi-ciphertext) security of FrodoKEM, addressing the multi-ciphertext attack described in Section 1.3. These results, reported in Section 7.1 and Appendix C, include bounds in the quantum random oracle model.
- In Section 6 and Section 7.2, we revisit the original FrodoKEM parameter sets and incorporate recent cryptanalytic developments to verify security level estimates for the three parameterizations Frodo-640, Frodo-976, and Frodo-1344.
- In Section 8, we provide updated implementations and performance measurements for FrodoKEM and eFrodoKEM on Intel x64 and ARM platforms, with an optimized C implementation taking advantage of AES-NI and AVX2 instructions on Intel x64. These implementations are available at https://github.com/microsoft/PQCrypto-LWEKE.

2 Background

This section defines the cryptographic primitives and security notions that are relevant to FrodoPKE and FrodoKEM. For the required mathematical background corresponding to the LWE problem and lattices, refer to Appendix A.

2.1 Notation

Vectors are denoted by bold lower-case letters (e.g., $\mathbf{a}, \mathbf{b}, \mathbf{v}$), and matrices are denoted by bold upper-case letters (e.g., $\mathbf{A}, \mathbf{B}, \mathbf{S}$). For a set D, the set of m-dimensional vectors with entries in D is denoted by D^m, and the set of m-by-n matrices with entries in D is denoted by $D^{m \times n}$. For an n-dimensional vector $\mathbf{v}$, its ith entry for $0 \leq i < n$ is denoted by $\mathbf{v}_i$. For an m-by-n matrix $\mathbf{A}$, its (i, j)th entry (i.e., the entry in the ith row and jth column) for $0 \leq i < m$ and $0 \leq j < n$ is denoted by $\mathbf{A}_{i,j}$, and its ith row is denoted by $\mathbf{A}_i = (\mathbf{A}_{i,0}, \mathbf{A}_{i,1}, \ldots, \mathbf{A}_{i,n-1})$. The transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^{\mathrm{T}}$.

An m-bit string $\mathbf{k} \in \{0,1\}^m$ is written as a vector over the set $\{0,1\}$. The ring of integers is denoted by $\mathbb{Z}$, and, for a positive integer q, the quotient ring of integers modulo q is denoted by $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$. For a probability distribution or randomized algorithm χ, the notation $e \leftarrow\!\!\$\ \chi$ denotes drawing a value e according to χ. The n-fold product distribution of χ with itself is denoted by χ^n. For a finite set S, the uniform distribution on S is denoted by $U(S)$. The floor of a real number a, i.e., the largest integer less than or equal to a, is denoted by $\lfloor a \rfloor$. The closest integer to a real number a (with ties broken upward) is denoted by $\lfloor a \rceil = \lfloor a + 1/2 \rfloor$. For a real vector $\mathbf{v} \in \mathbb{R}^n$, its Euclidean (i.e., ℓ_2) norm is denoted by $\|\mathbf{v}\|$. For two n-dimensional vectors $\mathbf{a}, \mathbf{b}$ over a ring R, their inner product is denoted by $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=0}^{n-1} \mathbf{a}_i \mathbf{b}_i \in R$. For a positive integer n, $[n]$ denotes the set $\{1, 2, \ldots, n\}$.

2.2 Cryptographic definitions

This section recalls definitions of cryptographic primitives that are relevant to FrodoKEM, along with their correctness and security notions.

Definition 1 (Key encapsulation mechanism)**.** A *key encapsulation mechanism* KEM is a tuple of algorithms (KeyGen, Encaps, Decaps) along with a finite keyspace $\mathcal{K}$:

- KeyGen() $\$\!\!\rightarrow (pk, sk)$: A probabilistic *key generation algorithm* that outputs a public key pk and a secret key sk.
- Encaps(pk) $\$\!\!\rightarrow (c, \mathbf{ss})$: A probabilistic *encapsulation algorithm* that takes as input a public key pk, and outputs an encapsulation c and a shared secret $\mathbf{ss} \in \mathcal{K}$. The encapsulation is sometimes called a ciphertext.

- $\text{Decaps}(c, sk) \to \mathbf{ss}'$: A (usually deterministic) *decapsulation algorithm* that takes as input an encapsulation c and a secret key sk, and outputs a shared secret $\mathbf{ss}' \in \mathcal{K}$.

The notion of δ-correctness gives a bound on the probability of a legitimate protocol execution producing different keys in encapsulation and decapsulation.

Definition 2 (δ-correctness for KEMs)**.** A key encapsulation mechanism KEM is *δ-correct* if $\Pr[\mathbf{ss}' \neq \mathbf{ss} : (pk, sk) \leftarrow_{\$} \text{KeyGen}(); (c, \mathbf{ss}) \leftarrow_{\$} \text{Encaps}(pk); \mathbf{ss}' \leftarrow \text{Decaps}(c, sk)] \leq \delta$.

The following defines indistinguishability under chosen-ciphertext attack ($\mathsf{IND\text{-}CCA}$) for a key encapsulation mechanism.

Definition 3 ($\mathsf{IND\text{-}CCA}$ for KEMs)**.** Let KEM be a key encapsulation mechanism, and let $\mathcal{A}$ be an algorithm. The security experiment for *indistinguishability under adaptive chosen ciphertext attack* ($\mathsf{IND\text{-}CCA2}$, or just $\mathsf{IND\text{-}CCA}$) of KEM is $\text{Exp}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A})$, as defined in Figure 1. The advantage[6] of $\mathcal{A}$ in the experiment is

$$\text{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A}) := 2 \cdot \left| \Pr\left[\text{Exp}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2} \right| .$$

Note that $\mathcal{A}$ can be a classical or quantum algorithm. Even if $\mathcal{A}$ is a quantum algorithm, we still require it to make classical queries to its $\mathcal{O}_{\text{Decaps}}$ oracle.

Experiment $\text{Exp}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A})$:

1: $(pk, sk) \leftarrow_{\$} \mathsf{KEM}.\text{KeyGen}()$
2: $b \leftarrow_{\$} \{0, 1\}$
3: $(c^*, \mathbf{ss}_0) \leftarrow_{\$} \mathsf{KEM}.\text{Encaps}(pk)$
4: $\mathbf{ss}_1 \leftarrow_{\$} U(\mathcal{K})$
5: $b' \leftarrow_{\$} \mathcal{A}^{\mathcal{O}_{\text{Decaps}}(\cdot)}(pk, \mathbf{ss}_b, c^*)$
6: **if** $b' = b$ **then**
7: **return** 1
8: **else**
9: **return** 0

Oracle $\mathcal{O}_{\text{Decaps}}(c)$:

1: **if** $c = c^*$ **then**
2: **return** $\perp$
3: **else**
4: **return** $\mathsf{KEM}.\text{Decaps}(c, sk)$

Figure 1: Security experiment for indistinguishability under adaptive chosen ciphertext attack ($\mathsf{IND\text{-}CCA2}$, or just $\mathsf{IND\text{-}CCA}$) of a key encapsulation mechanism KEM for an adversary $\mathcal{A}$.

The key encapsulation mechanism presented in this work is obtained by transforming a *public-key encryption* (PKE) scheme, which is formally defined as follows.

Definition 4 (Public-key encryption scheme)**.** A *public-key encryption scheme* PKE is a tuple of algorithms $(\text{KeyGen}, \text{Enc}, \text{Dec})$ along with a message space $\mathcal{M}$:

- $\text{KeyGen}() \xrightarrow{\$} (pk, sk)$: A probabilistic *key generation algorithm* that outputs a public key pk and a secret key sk.
- $\text{Enc}(m, pk) \xrightarrow{\$} c$: A probabilistic *encryption algorithm* that takes as input a message $m \in \mathcal{M}$ and public key pk, and outputs a ciphertext c. The deterministic form is denoted $\text{Enc}(m, pk; r) \to c$, where the randomness $r \in \mathcal{R}$ is passed as an explicit input; $\mathcal{R}$ is called the *randomness space* of the encryption algorithm.
- $\text{Dec}(c, sk) \to m'$ or $\perp$: A deterministic *decryption algorithm* that takes as input a ciphertext c and secret key sk, and outputs either a message $m' \in \mathcal{M}$ or a special error symbol $\perp \notin \mathcal{M}$.

[6] Note that we define advantage as $2 \cdot |\Pr[\ldots] - 1/2|$, whereas some FO literature defines it as $|\Pr[\ldots] - 1/2|$. Our definition normalizes the advantage to be in $[0, 1]$, and is consistent with the definition of statistical distance.

Some results rely on the unpredictabilty (min-entropy) of the ciphertext (taken over the randomness of the encryption algorithm), as captured by the following definition:

Definition 5 (γ-spreadness)**.** We say that a public-key encryption scheme PKE is *γ-spread* if for all key pairs (pk, sk) in the support of KeyGen and all messages $m \in \mathcal{M}$, it holds that $\max_c \Pr[\mathrm{Enc}(m, pk) = c] \leq 2^{-\gamma}$, where the probability is taken over the internal randomness of Enc.

The notion of δ-correctness captures an upper bound on the probability of decryption failure in a legitimate execution of the scheme.

Definition 6 (δ-correctness for PKEs [HHK17])**.** A public-key encryption scheme PKE with message space $\mathcal{M}$ is *δ-correct* if

$$\mathbb{E}\left[\max_{m \in \mathcal{M}} \Pr\left[\mathsf{PKE}.\mathrm{Dec}(c, sk) \neq m : c \leftarrow\!\!\$\ \mathsf{PKE}.\mathrm{Enc}(m, pk)\right]\right] \leq \delta \ , \tag{1}$$

where the expectation is taken over $(pk, sk) \leftarrow\!\!\$\ \mathsf{PKE}.\mathrm{KeyGen}()$.

In $\mathsf{FrodoPKE}$, the probability expression in Equation 1 has no dependence on m, so the condition simplifies to

$$\Pr\left[\mathsf{PKE}.\mathrm{Dec}(c, sk) \neq m : (pk, sk) \leftarrow \mathsf{PKE}.\mathrm{KeyGen}(); c \leftarrow\!\!\$\ \mathsf{PKE}.\mathrm{Enc}(m, pk)\right] \leq \delta \ , \tag{2}$$

which is what we analyze when calculating the probability of decryption failure (see Section 4.1).

The PKE scheme we use as the basis for the KEM transformation in Section 4.2 is required to satisfy the notion of $\mathsf{IND\text{-}CPA}$ security, which is defined as follows.

Definition 7 ($\mathsf{IND\text{-}CPA}$ for PKE)**.** Let PKE be a public-key encryption scheme, and let $\mathcal{A}$ be an algorithm. The security experiment for *indistinguishability under chosen plaintext attack* ($\mathsf{IND\text{-}CPA}$) of PKE is $\mathrm{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A})$ shown in Figure 2. The advantage of $\mathcal{A}$ in the experiment is

$$\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) := 2 \cdot \left| \Pr\left[\mathrm{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2} \right|.$$

Experiment $\mathrm{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A})$:

1: $(pk, sk) \leftarrow\!\!\$\ \mathsf{PKE}.\mathrm{KeyGen}()$
2: $(m_0, m_1, st) \leftarrow\!\!\$\ \mathcal{A}(pk)$
3: $b \leftarrow\!\!\$\ \{0, 1\}$
4: $c^* \leftarrow\!\!\$\ \mathsf{PKE}.\mathrm{Enc}(m_b, pk)$
5: $b' \leftarrow\!\!\$\ \mathcal{A}(pk, c^*, st)$
6: **if** $b' = b$ **then**
7: **return** 1
8: **else**
9: **return** 0

Figure 2: Security experiment for indistinguishability under chosen plaintext attack ($\mathsf{IND\text{-}CPA}$) of a public-key encryption scheme PKE against an adversary $\mathcal{A}$.

In the multi-key (i.e., multi-user) setting, we need a suitably adapted definition of correctness.

Definition 8 ($\delta(n_u)$-correctness for PKEs)**.** A public-key encryption scheme PKE with message space $\mathcal{M}$ is called $\delta(n_u)$-correct if for all n_u,

$$\mathbb{E}\left[\max_{j\in[n_u]}\max_{m\in\mathcal{M}}\Pr[\mathsf{PKE.Dec}(sk_j,\mathsf{PKE.Enc}(pk_j,m))\neq m]\right]\leq\delta(n_u)$$

where the expectation is taken over $(pk_1,sk_1),\ldots,(pk_{n_u},sk_{n_u})\leftarrow\!\!\$\ \mathsf{PKE.KeyGen}()$.

We note that in $\mathsf{FrodoPKE}$, the probability of incorrect decryption does not depend on the choice of message $m\in\mathcal{M}$, so the expression from Definition 8 simplifies to

$$\mathbb{E}\left[\max_{j\in[n_u]}\Pr[\mathsf{PKE.Dec}(sk_j,\mathsf{PKE.Enc}(pk_j,m))\neq m]\right],$$

where $m\in\mathcal{M}$ is some arbitrary message. Also note that by restricting to $n_u=1$, the above definitions simplify to merely single-user correctness [HHK17] (see Definition 6). Because the maximum of several non-negative values is at most their sum, by linearity of expectation, any δ-correct PKE is $\delta(n_u)$-correct for $\delta(n_u)=\delta\cdot n_u$. However, there is strong evidence that for $\mathsf{FrodoPKE}$ and other natural lattice-based schemes, $\delta(n_u)\ll\delta\cdot n_u$; see [DHK+21, Table 1].

In the multi-key security experiments, an oracle generates each ciphertext by encrypting under one of several (properly generated) public keys, as specified by the adversary. Formally, we define $\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}$ security for a PKE, and $\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}$ security for a KEM. Note that $\mathsf{IND\text{-}CPA}$ and $\mathsf{IND\text{-}CCA}$ are simply the special cases of these notions, respectively, for $n_c=n_u=1$. Our multi-target definitions differ slightly from those of [DHK+21], which allow the adversary to concentrate all its challenge queries on a single user; we follow the approach of Bellare et al. [BBM00], where the adversary is limited to n_c challenge queries per user, so the total number of challenges is bounded by $n_u\cdot n_c$.

Definition 9 ($\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}$ for PKE [GHS25,Gla24])**.** Let PKE be a public-key encryption scheme and let $\mathcal{A}$ be an algorithm. The $\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}$ security experiment for $\mathcal{A}$ attacking PKE is $\mathrm{Exp}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A})$ from Figure 3. The advantage of $\mathcal{A}$ in the experiment is

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A}):=2\cdot\left|\Pr\left[\mathrm{Exp}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A})\Rightarrow 1\right]-\frac{1}{2}\right|.$$

Experiment $\mathrm{Exp}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A})$:

$b\leftarrow\!\!\$\ \{0,1\}$
for $j=1,\ldots,n_u$ **do**
  $(pk_j,sk_j)\leftarrow\!\!\$\ \mathsf{PKE.KeyGen}()$
$\vec{pk}\leftarrow(pk_1,\ldots,pk_{n_u})$
$b'\leftarrow\!\!\$\ \mathcal{A}^{\mathsf{chall}_1,\ldots,\mathsf{chall}_{n_u}}(\vec{pk})$
if $b=b'$ **then**
  return 1
else
  return 0

Oracle $\mathsf{chall}_j(m_0,m_1)$:

[may be called up to n_c times]
$c\leftarrow\!\!\$\ \mathsf{PKE.Enc}(pk_j,m_b)$
return c

Figure 3: Security experiment for $\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}$ security of a public-key encryption scheme PKE against an adversary $\mathcal{A}$.

Definition 10 ($\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}$ for KEM [DHK+21])**.** Let KEM be a key encapsulation mechanism, and let $\mathcal{A}$ be an algorithm. The $\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}$ security experiment for $\mathcal{A}$

attacking KEM is $\mathrm{Exp}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{A})$ from Figure 4. The advantage of $\mathcal{A}$ in the experiment is

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{A}) := 2 \cdot \left| \Pr\left[\mathrm{Exp}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{A}) \Rightarrow 1\right] - \frac{1}{2} \right|.$$

Experiment $\mathrm{Exp}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{A})$:

$b \leftarrow\!\!\$\ \{0,1\}$
for $j = 1, \ldots, n_u$ **do**
  $(pk_j, sk_j) \leftarrow\!\!\$\ \mathsf{KEM}.\mathrm{KeyGen}()$
$\vec{pk} \leftarrow (pk_1, \ldots, pk_{n_u})$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_{\mathrm{Decaps}}, \mathsf{chall}_1, \ldots, \mathsf{chall}_{n_u}}(\vec{pk})$
if $b = b'$ **then**
  return 1
else
  return 0

Oracle $\mathcal{O}_{\mathrm{Decaps}}(j, c)$:

if $c \in \mathfrak{L}_{C_j}$ **then**
  return $\perp$
else
  return $\mathsf{KEM}.\mathrm{Decaps}(c, sk_j)$

Oracle $\mathsf{chall}_j()$:

[may be called up to n_c times]
$(c, k_0) \leftarrow\!\!\$\ \mathsf{KEM}.\mathrm{Encaps}(pk_j)$
$\mathfrak{L}_{C_j} \leftarrow \mathfrak{L}_{C_j} \cup c$
$k_1 \leftarrow\!\!\$\ \mathcal{K}$
return (c, k_b)

Figure 4: Security experiment for $\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CCA}$ security of a key encapsulation mechanism KEM against an adversary $\mathcal{A}$. Here $\mathfrak{L}_{C_j}$ denotes the list of ciphertexts encrypted under pk_j, and is initialized to be empty.

3 Auxiliary algorithms

This section describes the auxiliary algorithms used in FrodoPKE and FrodoKEM.

Notation. In this work, the algorithms are described in terms of the following parameters:

- χ, a probability distribution on $\mathbb{Z}$, and T_χ, the corresponding distribution table for sampling;
- $q = 2^D$, a power-of-two integer modulus with exponent $D \leq 16$;
- $n, \overline{m}, \overline{n}$, integer matrix dimensions with $n \equiv 0 \pmod 8$;
- $B \leq D$, the number of bits encoded in each matrix entry;
- $\ell = B \cdot \overline{m} \cdot \overline{n}$, the length of bit strings that are encoded as $\overline{m}$-by-$\overline{n}$ matrices;
- $\mathsf{len}_{\mathsf{seed}_\mathbf{A}}$, the bit length of seeds used for pseudorandom matrix generation;
- $\mathsf{len}_{\mathsf{seed}_\mathbf{SE}}$, the bit length of seeds used for pseudorandom bit generation for error sampling.

3.1 Sampling from the error distribution

The error distribution χ used in FrodoKEM is a discrete, symmetric distribution on $\mathbb{Z}$, centered at zero and with small support, which approximates a rounded continuous Gaussian distribution.

The support of χ is $S_\chi = \{-s, -s+1, \ldots, -1, 0, 1, \ldots, s-1, s\}$ for a positive integer s. The probabilities $\chi(z) = \chi(-z)$ for $z \in S_\chi$ are given by a discrete probability density function, which is described by a table $T_\chi = (T_\chi(0), T_\chi(1), \ldots, T_\chi(s))$ of $s+1$ positive integers related to the cumulative distribution function. For a certain positive integer len_χ, the table entries satisfy the following conditions:

$$T_\chi(0) = 2^{\mathsf{len}_\chi - 1} \cdot \chi(0) - 1 \qquad \text{and} \qquad T_\chi(z) = T_\chi(0) + 2^{\mathsf{len}_\chi} \sum_{i=1}^{z} \chi(i) \quad \text{for } 1 \leq z \leq s.$$

Since the distribution χ is symmetric and centered at zero, it is easy to verify that $T_\chi(s) = 2^{\mathsf{len}_\chi - 1} - 1$.

Sampling from χ via inversion sampling is done as shown in Algorithm 1. Given a string of len_χ uniformly random bits $\mathbf{r} \in \{0,1\}^{\mathsf{len}_\chi}$ and a distribution table T_χ, the algorithm Frodo.Sample returns a sample e from the distribution χ. (Note that $T_\chi(s)$ is never accessed.) We emphasize that it is important to perform this sampling in constant time to avoid exposing timing side-channels, which is why Step 3 of the algorithm does a complete loop through the entire table T_χ. The comparison in Step 4 needs to be implemented in a constant-time manner.

Algorithm 1 Frodo.Sample

Input: A (random) bit string $\mathbf{r} = (\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{\mathsf{len}_\chi - 1}) \in \{0,1\}^{\mathsf{len}_\chi}$, the table $T_\chi = (T_\chi(0), T_\chi(1), \ldots, T_\chi(s))$.
Output: A sample $e \in \mathbb{Z}$.

1: $t \leftarrow \sum_{i=1}^{\mathsf{len}_\chi - 1} \mathbf{r}_i \cdot 2^{i-1}$
2: $e \leftarrow 0$
3: **for** $(z = 0;\ z < s;\ z \leftarrow z + 1)$ **do**
4: **if** $t > T_\chi(z)$ **then**
5: $e \leftarrow e + 1$
6: $e \leftarrow (-1)^{\mathbf{r}_0} \cdot e$
7: **return** e

An n_1-by-n_2 matrix of $n_1 n_2$ samples from the error distribution is sampled on input of a $(n_1 n_2 \cdot \mathsf{len}_\chi)$-bit string, here written as a sequence $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(n_1 n_2 - 1)})$ of $n_1 n_2$ bit vectors of length len_χ each, by sampling $n_1 n_2$ error terms through calls to Frodo.Sample on a corresponding len_χ-bit substring $\mathbf{r}^{(i \cdot n_2 + j)}$ and the distribution table T_χ to sample the matrix entry $\mathbf{E}_{i,j}$. The algorithm Frodo.SampleMatrix is shown in Algorithm 2.

Algorithm 2 Frodo.SampleMatrix

Input: A (random) bit string $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(n_1 n_2 - 1)}) \in \{0,1\}^{n_1 n_2 \cdot \mathsf{len}_\chi}$ (here, each $\mathbf{r}^{(i)}$ is a vector of len_χ bits), the table T_χ.
Output: A sample $\mathbf{E} \in \mathbb{Z}^{n_1 \times n_2}$.

1: **for** $(i = 0;\ i < n_1;\ i \leftarrow i + 1)$ **do**
2: **for** $(j = 0;\ j < n_2;\ j \leftarrow j + 1)$ **do**
3: $\mathbf{E}_{i,j} \leftarrow$ Frodo.Sample$(\mathbf{r}^{(i \cdot n_2 + j)}, T_\chi)$
4: **return** $\mathbf{E}$

3.2 Pseudorandom matrix generation

The algorithm Frodo.Gen takes as input a seed $\mathsf{seed}_\mathbf{A} \in \{0,1\}^{\mathsf{len}_{\mathsf{seed}_\mathbf{A}}}$ and an implicit dimension $n \in \mathbb{Z}$, and outputs a pseudorandom matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$. There are two options for instantiating Frodo.Gen: using AES128 (Algorithm 3) and using SHAKE128 (Algorithm 4). Since the modulus q is a power of 2, reducing uniform bit strings modulo q introduces no bias.

3.3 Matrix encoding and packing

FrodoKEM requires functions to encode and decode bit strings as mod-q integer matrices, as well as packing and unpacking algorithms that transform matrices with entries in $\mathbb{Z}_q$ to bit strings and vice versa. The corresponding functions Frodo.Encode, Frodo.Decode, Frodo.Pack and Frodo.Unpack are detailed in Appendix B.

Algorithm 3 Frodo.Gen using AES128

Input: Seed $\mathsf{seed}_{\mathbf{A}} \in \{0,1\}^{\mathsf{len}_{\mathsf{seed}_{\mathbf{A}}}}$.
Output: Matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$.

1: **for** $(i = 0; i < n; i \leftarrow i + 1)$ **do**
2: **for** $(j = 0; j < n; j \leftarrow j + 8)$ **do**
3: $\mathbf{b} \leftarrow \langle i \rangle \| \langle j \rangle \| 0 \cdots 0 \in \{0,1\}^{128}$ where $\langle i \rangle, \langle j \rangle \in \{0,1\}^{16}$
4: $\langle c_{i,j} \rangle \| \langle c_{i,j+1} \rangle \| \cdots \| \langle c_{i,j+7} \rangle \leftarrow \mathrm{AES128}_{\mathsf{seed}_{\mathbf{A}}}(\mathbf{b})$ where each $\langle c_{i,k} \rangle \in \{0,1\}^{16}$
5: **for** $(k = 0; k < 8; k \leftarrow k + 1)$ **do**
6: $\mathbf{A}_{i,j+k} \leftarrow c_{i,j+k} \bmod q$
7: **return** $\mathbf{A}$

Algorithm 4 Frodo.Gen using SHAKE128

Input: Seed $\mathsf{seed}_{\mathbf{A}} \in \{0,1\}^{\mathsf{len}_{\mathsf{seed}_{\mathbf{A}}}}$.
Output: Pseudorandom matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$.

1: **for** $(i = 0; i < n; i \leftarrow i + 1)$ **do**
2: $\mathbf{b} \leftarrow \langle i \rangle \| \mathsf{seed}_{\mathbf{A}} \in \{0,1\}^{16 + \mathsf{len}_{\mathsf{seed}_{\mathbf{A}}}}$ where $\langle i \rangle \in \{0,1\}^{16}$
3: $\langle c_{i,0} \rangle \| \langle c_{i,1} \rangle \| \cdots \| \langle c_{i,n-1} \rangle \leftarrow \mathrm{SHAKE128}(\mathbf{b}, 16n)$ where each $\langle c_{i,j} \rangle \in \{0,1\}^{16}$
4: **for** $(j = 0; j < n; j \leftarrow j + 1)$ **do**
5: $\mathbf{A}_{i,j} \leftarrow c_{i,j} \bmod q$
6: **return** $\mathbf{A}$

3.4 Deterministic random bit generation

FrodoKEM requires the deterministic generation of random bit sequences from a random seed value. This is done using the SHA-3-derived extendable output function SHAKE [Dwo15]. The function SHAKE is taken as either SHAKE128 or SHAKE256 (indicated in Table 1 for each parameter set of FrodoKEM), and takes as input a bit string X and a requested output bit length L.

4 FrodoPKE: IND-CPA-secure public-key encryption

This section describes FrodoPKE, a public-key encryption scheme with fixed-length message space, targeting IND-CPA security, that will be used as a building block for FrodoKEM. FrodoPKE is based on the public-key encryption scheme by Lindner and Peikert [LP11].

The PKE scheme is given by three algorithms (FrodoPKE.KeyGen, FrodoPKE.Enc, FrodoPKE.Dec), defined respectively in Algorithm 5, Algorithm 6, and Algorithm 7. FrodoPKE is parameterized by the parameters defined in Section 3. Additional parameters include the bit length $\mathsf{len}_\mu = \ell$ of messages, the message space $\mathcal{M} = \{0,1\}^{\mathsf{len}_\mu}$, and the matrix-generation algorithm Frodo.Gen (either Algorithm 3 or Algorithm 4). In the notation of [LP11], their n_1 and n_2 both equal n here, and their dimension ℓ is $\overline{n}$ here.

4.1 Correctness of FrodoPKE

The next lemma states bounds on the size of errors that can be handled by the decoding algorithm.

Lemma 1. *Let* $\mathrm{ec}(\cdot)$ *and* $\mathrm{dc}(\cdot)$ *denote the encoding and decoding functions (resp.) defined in Appendix B. Let* $q = 2^D$, $B \leq D$. *Then* $\mathrm{dc}(\mathrm{ec}(k) + e) = k$ *for any* $k, e \in \mathbb{Z}$ *such that* $0 \leq k < 2^B$ *and* $-q/2^{B+1} \leq e < q/2^{B+1}$.

Proof. This follows directly from the fact that $\mathrm{dc}(\mathrm{ec}(k) + e) = \lfloor k + e2^B/q \rceil \bmod 2^B$. □

Algorithm 5 FrodoPKE.KeyGen.

Input: None.
Output: Key pair $(pk, sk) \in (\{0,1\}^{\mathsf{len}_{\mathsf{seed}_\mathbf{A}}} \times \mathbb{Z}_q^{n\times\overline{n}}) \times \mathbb{Z}_q^{\overline{n}\times n}$.

1: Choose a uniformly random seed $\mathsf{seed}_\mathbf{A} \leftarrow\!\!\$\ U(\{0,1\}^{\mathsf{len}_{\mathsf{seed}_\mathbf{A}}})$
2: Generate the matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$ via $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathsf{seed}_\mathbf{A})$
3: Choose a uniformly random seed $\mathsf{seed}_\mathbf{SE} \leftarrow\!\!\$\ U(\{0,1\}^{\mathsf{len}_{\mathsf{seed}_\mathbf{SE}}})$
4: Generate pseudorandom bit string $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(2n\overline{n}-1)}) \leftarrow \mathrm{SHAKE}(\mathtt{0x5F}\|\mathsf{seed}_\mathbf{SE}, 2n\overline{n} \cdot \mathsf{len}_\chi)$
5: Sample error matrix $\mathbf{S}^\mathrm{T} \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(n\overline{n}-1)}), \overline{n}, n, T_\chi)$
6: Sample error matrix $\mathbf{E} \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(n\overline{n})}, \mathbf{r}^{(n\overline{n}+1)}, \ldots, \mathbf{r}^{(2n\overline{n}-1)}), n, \overline{n}, T_\chi)$
7: Compute $\mathbf{B} = \mathbf{AS} + \mathbf{E}$
8: **return** public key $pk \leftarrow (\mathsf{seed}_\mathbf{A}, \mathbf{B})$ and secret key $sk \leftarrow \mathbf{S}^\mathrm{T}$

Algorithm 6 FrodoPKE.Enc.

Input: Message $\mu \in \mathcal{M}$ and public key $pk = (\mathsf{seed}_\mathbf{A}, \mathbf{B}) \in \{0,1\}^{\mathsf{len}_{\mathsf{seed}_\mathbf{A}}} \times \mathbb{Z}_q^{n\times\overline{n}}$.
Output: Ciphertext $c = (\mathbf{C}_1, \mathbf{C}_2) \in \mathbb{Z}_q^{\overline{m}\times n} \times \mathbb{Z}_q^{\overline{m}\times\overline{n}}$.

1: Generate $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathsf{seed}_\mathbf{A})$
2: Choose a uniformly random seed $\mathsf{seed}_\mathbf{SE} \leftarrow\!\!\$\ U(\{0,1\}^{\mathsf{len}_{\mathsf{seed}_\mathbf{SE}}})$
3: Generate pseudorandom bit string $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(2\overline{m}n+\overline{m}\overline{n}-1)}) \leftarrow \mathrm{SHAKE}(\mathtt{0x96}\|\mathsf{seed}_\mathbf{SE}, (2\overline{m}n + \overline{m}\overline{n}) \cdot \mathsf{len}_\chi)$
4: Sample error matrix $\mathbf{S}' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(\overline{m}n-1)}), \overline{m}, n, T_\chi)$
5: Sample error matrix $\mathbf{E}' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(\overline{m}n)}, \mathbf{r}^{(\overline{m}n+1)}, \ldots, \mathbf{r}^{(2\overline{m}n-1)}), \overline{m}, n, T_\chi)$
6: Sample error matrix $\mathbf{E}'' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(2\overline{m}n)}, \mathbf{r}^{(2\overline{m}n+1)}, \ldots, \mathbf{r}^{(2\overline{m}n+\overline{m}\overline{n}-1)}), \overline{m}, \overline{n}, T_\chi)$
7: Compute $\mathbf{B}' = \mathbf{S}'\mathbf{A} + \mathbf{E}'$ and $\mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}''$
8: **return** ciphertext $c \leftarrow (\mathbf{C}_1, \mathbf{C}_2) = (\mathbf{B}', \mathbf{V} + \mathsf{Frodo.Encode}(\mu))$

Algorithm 7 FrodoPKE.Dec.

Input: Ciphertext $c = (\mathbf{C}_1, \mathbf{C}_2) \in \mathbb{Z}_q^{\overline{m}\times n} \times \mathbb{Z}_q^{\overline{m}\times\overline{n}}$ and secret key $sk = \mathbf{S}^\mathrm{T} \in \mathbb{Z}_q^{\overline{n}\times n}$.
Output: Decrypted message $\mu' \in \mathcal{M}$.

1: Compute $\mathbf{M} = \mathbf{C}_2 - \mathbf{C}_1\mathbf{S}$
2: **return** message $\mu' \leftarrow \mathsf{Frodo.Decode}(\mathbf{M})$

Correctness of decryption: The decryption algorithm FrodoPKE.Dec computes

$$\begin{aligned}
\mathbf{M} &= \mathbf{C}_2 - \mathbf{C}_1\mathbf{S} \\
&= \mathbf{V} + \mathsf{Frodo.Encode}(\mu) - (\mathbf{S}'\mathbf{A} + \mathbf{E}')\mathbf{S} \\
&= \mathsf{Frodo.Encode}(\mu) + \mathbf{S}'\mathbf{B} + \mathbf{E}'' - \mathbf{S}'\mathbf{AS} - \mathbf{E}'\mathbf{S} \\
&= \mathsf{Frodo.Encode}(\mu) + \mathbf{S}'\mathbf{AS} + \mathbf{S}'\mathbf{E} + \mathbf{E}'' - \mathbf{S}'\mathbf{AS} - \mathbf{E}'\mathbf{S} \\
&= \mathsf{Frodo.Encode}(\mu) + \mathbf{S}'\mathbf{E} + \mathbf{E}'' - \mathbf{E}'\mathbf{S} \\
&= \mathsf{Frodo.Encode}(\mu) + \mathbf{E}'''
\end{aligned}$$

for some error matrix $\mathbf{E}''' = \mathbf{S}'\mathbf{E} + \mathbf{E}'' - \mathbf{E}'\mathbf{S}$. Therefore, any B-bit substring of the message μ corresponding to an entry of $\mathbf{M}$ will be decrypted correctly if the condition in Lemma 1 is satisfied for the corresponding entry of $\mathbf{E}'''$.

Failure probability. Each entry in the matrix $\mathbf{E}'''$ is the sum of $2n$ products of two independent samples from χ, and one more independent sample from χ. Denote the distribution of this sum by χ'. In the case of a power-of-2 modulus q, the probability of decryption failure for any single symbol is therefore the sum $p = \sum_{e \notin [-q/2^{B+1}, q/2^{B+1})} \chi'(e)$. The probability of decryption failure for the entire message can then be obtained using the union bound.

For the distributions χ we use, which have rather small support S_χ, the distribution χ' can be efficiently computed exactly. The probability that a product of two independent samples from χ equals e (modulo q) is simply $\sum_{(a,b) \in S_\chi \times S_\chi \,:\, ab = e \bmod q} \chi(a) \cdot \chi(b)$. Similarly, the probability that the sum of two entries assumes a certain value is given by the standard convolution sum. Section 6.2 reports the failure probability for each of the selected parameter sets.

4.2 Transform from IND_{n_c,n_u}-CPA PKE to IND_{n_c,n_u}-CCA KEM

The Fujisaki–Okamoto transform [FO99b] constructs an IND-CCA2-secure public-key encryption scheme, in the classical random oracle model, from a one-way-secure public-key encryption scheme (assuming the distribution of ciphertexts for each plaintext is sufficiently "well spread"). Targhi and Unruh [TU16] gave a variant of the Fujisaki–Okamoto transform and proved its IND-CCA2 security against a quantum adversary in the quantum random oracle model under similar assumptions. The results of both [FO99b] and [TU16] proceed under the assumption that the public-key encryption scheme has perfect correctness, which is often not the case for lattice-based schemes (including ours). Hofheinz, Hövelmanns, and Kiltz [HHK17] gave a variety of constructions, in a modular fashion, that in particular allow for a small probability of incorrect decryption.

The FrodoKEM 2023 "annex" update document [ABD+23] adapted the $\mathsf{FO}^{\not\perp}$ transform from [HHK17] and proposed a variant called the Modified Salted Fujisaki–Okamoto with implicit rejection ($\mathsf{SFO}^{\not\perp\prime}$) transform. In this transform, encapsulation generates a uniformly random, public salt of bit length $\ell = \mathsf{len}_{\mathsf{salt}}$, and includes it in the output ciphertext. The security of the resulting KEM against multi-target attacks in the classical ROM was proved in [GHS25] (and a QROM bound was also proved). Without the salt, and with n_c challenge ciphertexts and N ROM queries, an adversary can break IND_{n_c,n_u}-CCA security with advantage roughly $n_c N/|\mathcal{M}|$. By including a salt of suitable length, the advantage is bounded according to Theorem 1 below.

FrodoKEM uses the $\mathsf{SFO}^{\not\perp\prime}$ transform, which constructs an IND_{n_c,n_u}-CCA-secure key encapsulation mechanism from an IND_{n_c,n_u}-CPA public-key encryption scheme and three hash functions; following [BDK+18], the transform also includes the following modifications (see Figure 5 for notation):

- A single hash function (with longer output) is used to compute $\mathbf{r}$ and $\mathbf{k}$.
- The computation of $\mathbf{r}$ and $\mathbf{k}$ also takes the public key pk as input.

Definition 11 ($\mathsf{SFO}^{\not\perp\prime}$ transform)**.** Let $\mathsf{PKE} = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ be a public-key encryption scheme with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$, where the randomness space of Enc is $\mathcal{R}$. Let $\mathsf{len}_{\mathsf{salt}}, \mathsf{len}_{\mathbf{s}}, \mathsf{len}_{\mathbf{k}}, \mathsf{len}_{\mathbf{pkh}}, \mathsf{len}_{\mathbf{ss}}$ be parameters. Let $G_1 \colon \{0,1\}^* \to \{0,1\}^{\mathsf{len}_{\mathbf{pkh}}}$, $G_2 \colon \{0,1\}^* \to \mathcal{R} \times \{0,1\}^{\mathsf{len}_{\mathbf{k}}}$, and $F \colon \{0,1\}^* \to \{0,1\}^{\mathsf{len}_{\mathbf{ss}}}$ be hash functions. Define $\mathsf{KEM}^{\not\perp\prime} = \mathsf{SFO}^{\not\perp\prime}[\mathsf{PKE}, G_1, G_2, F]$ to be the key encapsulation mechanism as shown in Figure 5.

As observed by Guo, Johansson, and Nilsson [GJN20], a timing side-channel enables key recovery if Step 5 of $\mathsf{KEM}^{\not\perp\prime}$.Decaps is not performed in constant time.

Remark 1. If salt is removed from the $\mathsf{SFO}^{\not\perp\prime}$ transform in Figure 5, the transform becomes equivalent to the $\mathsf{FO}^{\not\perp\prime}$ transform [BDK+18, NAB+20]. That transform is used in our eFrodoKEM variant, which is intended for applications where only a small number of ciphertexts are generated per public key (see Section 5 and Section 7.1).

$\mathsf{KEM}^{\not\perp\prime}$.KeyGen():

1: $(pk, sk) \leftarrow_\$ \mathsf{PKE.KeyGen}()$
2: $\mathbf{s} \leftarrow_\$ \{0,1\}^{\mathsf{len_s}}$
3: $\mathbf{pkh} \leftarrow G_1(pk)$
4: $sk' \leftarrow (sk, \mathbf{s}, pk, \mathbf{pkh})$
5: **return** (pk, sk')

$\mathsf{KEM}^{\not\perp\prime}$.Encaps($pk$):

1: $\mu \leftarrow_\$ \mathcal{M}$, $\mathsf{salt} \leftarrow_\$ \{0,1\}^{\mathsf{len_{salt}}}$
2: $(\mathbf{r}, \mathbf{k}) \leftarrow G_2(G_1(pk) \| \mu \| \mathsf{salt})$
3: $c \leftarrow \mathsf{PKE.Enc}(\mu, pk; \mathbf{r})$
4: $\mathbf{ss} \leftarrow F(c \| \mathsf{salt} \| \mathbf{k})$
5: **return** $(c \| \mathsf{salt}, \mathbf{ss})$

$\mathsf{KEM}^{\not\perp\prime}$.Decaps($c \| \mathsf{salt}, (sk, \mathbf{s}, pk, \mathbf{pkh})$):

1: $\mu' \leftarrow \mathsf{PKE.Dec}(c, sk)$
2: $(\mathbf{r}', \mathbf{k}') \leftarrow G_2(\mathbf{pkh} \| \mu' \| \mathsf{salt})$
3: $\mathbf{ss}'_0 \leftarrow F(c \| \mathsf{salt} \| \mathbf{k}')$
4: $\mathbf{ss}'_1 \leftarrow F(c \| \mathsf{salt} \| \mathbf{s})$
5: (in constant time) $\mathbf{ss}' \leftarrow \mathbf{ss}'_0$ if $c = \mathsf{PKE.Enc}(\mu', pk; \mathbf{r}')$ else $\mathbf{ss}' \leftarrow \mathbf{ss}'_1$
6: **return** $\mathbf{ss}'$

Figure 5: Construction of an IND-CCA-secure key encapsulation mechanism $\mathsf{KEM}^{\not\perp\prime} = \mathsf{SFO}^{\not\perp\prime}[\mathsf{PKE}, G_1, G_2, F]$ from a public-key encryption scheme PKE and hash functions G_1, G_2, and F.

5 FrodoKEM: IND-CCA-secure key encapsulation

This section defines FrodoKEM, a key encapsulation mechanism that is derived from FrodoPKE by applying the $\mathsf{SFO}^{\not\perp\prime}$ transform. The KEM scheme is given by three algorithms (FrodoKEM.KeyGen, FrodoKEM.Encaps, FrodoKEM.Decaps), defined respectively in Algorithm 8, Algorithm 9, and Algorithm 10. FrodoKEM is parameterized by the following:

- $q = 2^D$, a power-of-two integer modulus with exponent $D \leq 16$;
- $n, \overline{m}, \overline{n}$, integer matrix dimensions with $n \equiv 0 \pmod{16}$;
- $B \leq D$, the number of bits encoded in each matrix entry;
- $\ell = B \cdot \overline{m} \cdot \overline{n}$, the length of bit strings to be encoded in an $\overline{m}$-by-$\overline{n}$ matrix;
- $\mathsf{len}_\mu = \ell$, the bit length of messages;
- $\mathcal{M} = \{0,1\}^{\mathsf{len}_\mu}$, the message space;
- $\mathsf{len}_{\mathsf{seed_A}}$, the bit length of seeds used for pseudorandom matrix generation;
- $\mathsf{len}_{\mathsf{seed_{SE}}}$, the bit length of seeds used for pseudorandom bit generation for error sampling;
- Frodo.Gen, pseudorandom matrix generation algorithm, either Algorithm 3 or Algorithm 4;
- T_χ, distribution table for sampling;
- $\mathsf{len_s}$, the length of the bit vector $\mathbf{s}$ used for pseudorandom shared secret generation in the event of decapsulation failure in the $\mathsf{SFO}^{\not\perp\prime}$ transform;
- $\mathsf{len_z}$, the bit length of seeds used for pseudorandom generation of $\mathsf{seed_A}$;
- $\mathsf{len_{salt}}$, the bit length of salt;
- $\mathsf{len_k}$, the bit length of intermediate shared secret $\mathbf{k}$ in the $\mathsf{SFO}^{\not\perp\prime}$ transform;
- $\mathsf{len_{pkh}}$, the bit length of the hash $G_1(pk)$ of the public key in the $\mathsf{SFO}^{\not\perp\prime}$ transform;
- $\mathsf{len_{ss}}$, the bit length of shared secret $\mathbf{ss}$ in the $\mathsf{SFO}^{\not\perp\prime}$ transform;

Correctness of IND-CCA KEM. For any KEM obtained from the FO transform, a correctness error occurs only on messages that exhibit a decryption error. Therefore, the failure probability δ of FrodoKEM is the same as the failure probability of the underlying FrodoPKE as computed in Section 4.1.

Algorithm 8 FrodoKEM.KeyGen.

Input: None.
Output: Key pair (pk, sk') with $pk \in \{0,1\}^{\mathsf{len}_{\mathsf{seed}_\mathbf{A}}+D\cdot n\cdot\overline{n}}$, $sk' \in \{0,1\}^{\mathsf{len}_\mathbf{s}+\mathsf{len}_{\mathsf{seed}_\mathbf{A}}+D\cdot n\cdot\overline{n}} \times \mathbb{Z}_q^{\overline{n}\times n} \times \{0,1\}^{\mathsf{len}_{\mathbf{pkh}}}$.

1: Choose uniformly random seeds $\mathbf{s}\|\mathsf{seed}_{\mathbf{SE}}\|\mathbf{z} \leftarrow_\$ U(\{0,1\}^{\mathsf{len}_\mathbf{s}+\mathsf{len}_{\mathsf{seed}_{\mathbf{SE}}}+\mathsf{len}_\mathbf{z}})$
2: Generate pseudorandom seed $\mathsf{seed}_\mathbf{A} \leftarrow \mathrm{SHAKE}(\mathbf{z}, \mathsf{len}_{\mathsf{seed}_\mathbf{A}})$
3: Generate the matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$ via $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathsf{seed}_\mathbf{A})$
4: Generate pseudorandom bit string $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(2n\overline{n}-1)}) \leftarrow \mathrm{SHAKE}(\mathtt{0x5F}\|\mathsf{seed}_{\mathbf{SE}}, 2n\overline{n}\cdot\mathsf{len}_\chi)$
5: Sample error matrix $\mathbf{S}^\mathrm{T} \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(n\overline{n}-1)}), \overline{n}, n, T_\chi)$
6: Sample error matrix $\mathbf{E} \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(n\overline{n})}, \mathbf{r}^{(n\overline{n}+1)}, \ldots, \mathbf{r}^{(2n\overline{n}-1)}), n, \overline{n}, T_\chi)$
7: Compute $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$
8: Compute $\mathbf{b} \leftarrow \mathsf{Frodo.Pack}(\mathbf{B})$
9: Compute $\mathbf{pkh} \leftarrow \mathrm{SHAKE}(\mathsf{seed}_\mathbf{A}\|\mathbf{b}, \mathsf{len}_{\mathbf{pkh}})$
10: **return** public key $pk \leftarrow \mathsf{seed}_\mathbf{A}\|\mathbf{b}$ and secret key $sk' \leftarrow (\mathbf{s}\|\mathsf{seed}_\mathbf{A}\|\mathbf{b}, \mathbf{S}^\mathrm{T}, \mathbf{pkh})$

Algorithm 9 FrodoKEM.Encaps.

Input: Public key $pk = \mathsf{seed}_\mathbf{A}\|\mathbf{b} \in \{0,1\}^{\mathsf{len}_{\mathsf{seed}_\mathbf{A}}+D\cdot n\cdot\overline{n}}$.
Output: Ciphertext $\mathbf{c}_1\|\mathbf{c}_2\|\mathsf{salt} \in \{0,1\}^{(\overline{m}\cdot n+\overline{m}\cdot\overline{n})D+\mathsf{len}_{\mathsf{salt}}}$ and shared secret $\mathbf{ss} \in \{0,1\}^{\mathsf{len}_{\mathbf{ss}}}$.

1: Choose uniformly random values $\mu \leftarrow_\$ U(\{0,1\}^{\mathsf{len}_\mu})$ and $\mathsf{salt} \leftarrow_\$ U(\{0,1\}^{\mathsf{len}_{\mathsf{salt}}})$
2: Compute $\mathbf{pkh} \leftarrow \mathrm{SHAKE}(pk, \mathsf{len}_{\mathbf{pkh}})$
3: Generate pseudorandom values $\mathsf{seed}_{\mathbf{SE}}\|\mathbf{k} \leftarrow \mathrm{SHAKE}(\mathbf{pkh}\|\mu\|\mathsf{salt}, \mathsf{len}_{\mathsf{seed}_{\mathbf{SE}}} + \mathsf{len}_\mathbf{k})$
4: Generate pseudorandom bit string $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(2\overline{m}n+\overline{m}\overline{n}-1)}) \leftarrow \mathrm{SHAKE}(\mathtt{0x96}\|\mathsf{seed}_{\mathbf{SE}}, (2\overline{m}n+\overline{m}\overline{n})\cdot\mathsf{len}_\chi)$
5: Sample error matrix $\mathbf{S}' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(\overline{m}n-1)}), \overline{m}, n, T_\chi)$
6: Sample error matrix $\mathbf{E}' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(\overline{m}n)}, \mathbf{r}^{(\overline{m}n+1)}, \ldots, \mathbf{r}^{(2\overline{m}n-1)}), \overline{m}, n, T_\chi)$
7: Generate $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathsf{seed}_\mathbf{A})$
8: Compute $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$
9: Compute $\mathbf{c}_1 \leftarrow \mathsf{Frodo.Pack}(\mathbf{B}')$
10: Sample error matrix $\mathbf{E}'' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(2\overline{m}n)}, \mathbf{r}^{(2\overline{m}n+1)}, \ldots, \mathbf{r}^{(2\overline{m}n+\overline{m}\overline{n}-1)}), \overline{m}, \overline{n}, T_\chi)$
11: Compute $\mathbf{B} \leftarrow \mathsf{Frodo.Unpack}(\mathbf{b}, n, \overline{n})$
12: Compute $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
13: Compute $\mathbf{C} \leftarrow \mathbf{V} + \mathsf{Frodo.Encode}(\mu)$
14: Compute $\mathbf{c}_2 \leftarrow \mathsf{Frodo.Pack}(\mathbf{C})$
15: Compute $\mathbf{ss} \leftarrow \mathrm{SHAKE}(\mathbf{c}_1\|\mathbf{c}_2\|\mathsf{salt}\|\mathbf{k}, \mathsf{len}_{\mathbf{ss}})$
16: **return** ciphertext $\mathbf{c}_1\|\mathbf{c}_2\|\mathsf{salt}$ and shared secret $\mathbf{ss}$

5.1 Cryptographic primitives

In FrodoKEM we use the following generic cryptographic primitives. We describe their security requirements and instantiations with NIST-approved cryptographic primitives.

- Frodo.Gen in FrodoKEM.KeyGen: The security requirement on Frodo.Gen is that it is a public function that generates pseudorandom matrices $\mathbf{A}$; see Section C.1.3. Frodo.Gen is instantiated using either AES128 (Algorithm 3) or SHAKE128 (Algorithm 4).
- G_1, G_2, and F in transform $\mathsf{SFO}^{\not\perp\prime}$: these are modeled as independent random oracles (see below). We instantiate these using either SHAKE128 or SHAKE256 as indicated

Algorithm 10 FrodoKEM.Decaps.

Input: Ciphertext $\mathbf{c}_1\|\mathbf{c}_2\|\mathsf{salt} \in \{0,1\}^{(\overline{m}\cdot n+\overline{m}\cdot\overline{n})D+\mathsf{len}_{\mathsf{salt}}}$, secret key $sk' = (\mathbf{s}\|\mathsf{seed}_{\mathbf{A}}\|\mathbf{b}, \mathbf{S}^{\mathrm{T}}, \mathbf{pkh}) \in \{0,1\}^{\mathsf{len}_{\mathbf{s}}+\mathsf{len}_{\mathsf{seed}_{\mathbf{A}}}+D\cdot n\cdot\overline{n}} \times \mathbb{Z}_q^{\overline{n}\times n} \times \{0,1\}^{\mathsf{len}_{\mathbf{pkh}}}$.
Output: Shared secret $\mathbf{ss} \in \{0,1\}^{\mathsf{len}_{\mathbf{ss}}}$.

1: $\mathbf{B}' \leftarrow$ Frodo.Unpack$(\mathbf{c}_1, \overline{m}, n)$
2: $\mathbf{C} \leftarrow$ Frodo.Unpack$(\mathbf{c}_2, \overline{m}, \overline{n})$
3: Compute $\mathbf{M} \leftarrow \mathbf{C} - \mathbf{B}'\mathbf{S}$
4: Compute $\mu' \leftarrow$ Frodo.Decode$(\mathbf{M})$
5: Parse $pk \leftarrow \mathsf{seed}_{\mathbf{A}}\|\mathbf{b}$
6: Generate pseudorandom values $\mathsf{seed}_{\mathbf{SE}}{}'\|\mathbf{k}' \leftarrow$ SHAKE$(\mathbf{pkh}\|\mu'\|\mathsf{salt}, \mathsf{len}_{\mathsf{seed}_{\mathbf{SE}}} + \mathsf{len}_{\mathbf{k}})$
7: Generate pseudorandom bit string $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(2\overline{m}n+\overline{m}\overline{n}-1)}) \leftarrow$ SHAKE$(\mathtt{0x96}\|\mathsf{seed}_{\mathbf{SE}}{}', (2\overline{m}n + \overline{m}\overline{n}) \cdot \mathsf{len}_{\chi})$
8: Sample error matrix $\mathbf{S}' \leftarrow$ Frodo.SampleMatrix$((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(\overline{m}n-1)}), \overline{m}, n, T_\chi)$
9: Sample error matrix $\mathbf{E}' \leftarrow$ Frodo.SampleMatrix$((\mathbf{r}^{(\overline{m}n)}, \mathbf{r}^{(\overline{m}n+1)}, \ldots, \mathbf{r}^{(2\overline{m}n-1)}), \overline{m}, n, T_\chi)$
10: Generate $\mathbf{A} \leftarrow$ Frodo.Gen$(\mathsf{seed}_{\mathbf{A}})$
11: Compute $\mathbf{B}'' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$
12: Sample error matrix $\mathbf{E}'' \leftarrow$ Frodo.SampleMatrix$((\mathbf{r}^{(2\overline{m}n)}, \mathbf{r}^{(2\overline{m}n+1)}, \ldots, \mathbf{r}^{(2\overline{m}n+\overline{m}\overline{n}-1)}), \overline{m}, \overline{n}, T_\chi)$
13: Compute $\mathbf{B} \leftarrow$ Frodo.Unpack$(\mathbf{b}, n, \overline{n})$
14: Compute $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
15: Compute $\mathbf{C}' \leftarrow \mathbf{V} +$ Frodo.Encode(μ')
16: (in constant time) $\overline{\mathbf{k}} \leftarrow \mathbf{k}'$ if $(\mathbf{B}'\|\mathbf{C} = \mathbf{B}''\|\mathbf{C}')$ else $\overline{\mathbf{k}} \leftarrow \mathbf{s}$
17: Compute $\mathbf{ss} \leftarrow$ SHAKE$(\mathbf{c}_1\|\mathbf{c}_2\|\mathsf{salt}\|\overline{\mathbf{k}}, \mathsf{len}_{\mathbf{ss}})$
18: **return** shared secret $\mathbf{ss}$

in Table 1 for each parameter set.

Domain separation for SHAKE**.** Each distinct use of SHAKE in FrodoKEM should be cryptographically independent, which is achieved via one of two forms of domain separation.

For uses of SHAKE where the inputs are of different lengths, we rely on Keccak's internal padding for domain separation, which pads the input string to a multiple of the rate using the string $\mathtt{10^*1}$.

For uses of SHAKE with inputs of the same length (i.e., Step 4 in FrodoKEM.KeyGen and FrodoKEM.Encaps, and Step 7 in FrodoKEM.Decaps), we prepend distinct bytes as domain separators. These domain separators have bit patterns ($\mathtt{0x5F} = \mathtt{01011111}$, $\mathtt{0x96} = \mathtt{10010110}$) that were chosen to make it hard to use individual or consecutive bit-flipping attacks to turn one into the other.

5.2 FrodoKEM variants

FrodoKEM is parameterized by the pseudorandom generator (PRG) that is used for the generation of the matrix $\mathbf{A}$ in Frodo.Gen. As explained in Section 3.2, there are two options: using AES128 (Algorithm 3) and using SHAKE128 (Algorithm 4).

In addition, FrodoKEM consists of two main variants: an "ephemeral" variant, called eFrodoKEM, that is intended for applications in which the number of ciphertexts produced relative to any single public key is fairly small (e.g., less than 2^8), and a "salted" variant, simply called FrodoKEM, that does not impose any restriction on the reuse of key pairs.

In contrast to eFrodoKEM, the salted KEM FrodoKEM is constructed by applying the $\mathsf{SFO}^{\not\perp\prime}$ transform and incorporates some changes to protect against multi-ciphertext

attacks; see Section 4.2. Specifically, salted FrodoKEM doubles the length of the $\mathsf{seed}_{\mathbf{SE}}$ value and incorporates a public random value salt into encapsulation (see Table 2).

6 Parameters

We propose the same parameter sets for FrodoPKE as those in the round-3 NIST submission [NAB+20], i.e., the concrete parameter sets remain unchanged. They were originally selected by a process framed as a combinatorial optimization problem, where the objective function was the ciphertext size, and the constraints were dictated by the target security level, probability of decryption failure, and computational efficiency. The optimization problem was solved by sweeping the parameter space, subject to simple pruning techniques.[7]

6.1 Parameter constraints

This section recalls the constraints for the original parameter selection in [NAB+20]. Implementation considerations limit q to be at most 2^{16} and n to be a multiple of 16. The sum of the bit lengths of FrodoPKE's ciphertext and its public key, which is $D \cdot (n \cdot (\overline{m} \ + \overline{n}) + \overline{m} \ \overline{n}) + \mathsf{len}_{\mathsf{seed}_{\mathbf{A}}}$, is used as the cost function.

The width of the Gaussian error distribution is taken to exceed the *smoothing parameter* [MR07] $\eta_\varepsilon(\mathbb{Z})$ of the integers, for a very small $\varepsilon > 0$. The specific values of σ are chosen following the methodology in Section C.1.5, which demonstrates that these choices conform to a nontrivial reduction from the worst-case BDDwDGS problem to the corresponding average-case LWE decision problem.

The complexity of the error-sampling algorithm (Section 3.1) depends on the support of the distribution and the number of uniformly random bits per sample, which is bounded by 16. Since the distribution is symmetric, the sample's sign ($\mathbf{r}_0$ in Algorithm 1) can be chosen independently from its magnitude e, which leaves 15 bits for sampling from the non-negative part of the support. For each setting of the variance σ^2, a discrete distribution is found subject to the above constraints that minimizes its Rényi divergence (for several integral orders) from the target "ideal" distribution, which is the rounded Gaussian $\Psi_{\sigma\sqrt{2\pi}}$.

The concrete security of the parameter sets was originally estimated based on cryptanalytic attacks as outlined in [NAB+20], closely following the methodology in [SAB+20] accounting for the loss due to substitution of a rounded Gaussian with its discrete approximation (Section 7.1.1). The probability of decryption failure was computed according to the procedure outlined in Section 4.

In case of ties, i.e., when different parameter sets resulted in identical ciphertext sizes (i.e., the same q and n), the smaller σ was chosen for FrodoKEM-640 and FrodoKEM-1344 (minimizing the probability of decryption failure), and the larger σ for FrodoKEM-976 (prioritizing security).

We renew the analysis of concrete cryptanalytic attacks and their cost estimation in Section 7.2 to account for cryptanalytic results that appeared in the meantime and by using a different cost estimator. We conclude that all three parameter sets can still be recommended at their intended security levels.

6.2 Selected parameter sets

The three core parameter sets for FrodoKEM are:

- Frodo-640, matching or exceeding the brute-force security of AES128,
- Frodo-976, matching or exceeding the brute-force security of AES192, and
- Frodo-1344, matching or exceeding the brute-force security of AES256,

[7] The original Python scripts accompanied the round-3 NIST submission (folder `Parameter_Search_Scripts`): https://frodokem.org/files/FrodoKEM-20200930.zip.

which target Levels 1, 3 and 5, respectively, in the NIST call for proposals [Nat17].

We parameterize each core set by the PRG that is used for the generation of matrix $\mathbf{A}$. As described in Section 3.2, FrodoKEM allows two options for the PRG: AES128 and SHAKE128. In addition, FrodoKEM consists of two main variants: a *salted* variant simply called FrodoKEM that does not impose any restriction on the reuse of key pairs, and an *ephemeral* variant called eFrodoKEM that is intended for applications in which the number of ciphertexts produced relative to any single public key is small (see Section 5.2).

Thus, in total, we propose twelve parameter sets: the variant FrodoKEM includes the parameter sets FrodoKEM-640-AES, FrodoKEM-976-AES, FrodoKEM-1344-AES, FrodoKEM-640-SHAKE, FrodoKEM-976-SHAKE and FrodoKEM-1344-SHAKE; the variant eFrodoKEM includes the parameter sets eFrodoKEM-640-AES, eFrodoKEM-976-AES, eFrodoKEM-1344-AES, eFrodoKEM-640-SHAKE, eFrodoKEM-976-SHAKE and eFrodoKEM-1344-SHAKE.

Table 1 and Table 2 summarize the cryptographic parameters for all the parameter sets. The corresponding error distributions appear in Table 3. Table 4 summarizes security claims we can make about FrodoKEM and its components. The columns under IND-CPA and IND-CCA security denote security, in bits, for estimates using the beyond-core-SVP method under the C-LSF-Sieve cost model (B), using the core-SVP method under the C-LSF-Sieve cost model (C), and using the core-SVP method under the Q-RW-Sieve cost model (Q) on $\overline{m}+\overline{n}$ instances of the normal-form (decisional) LWE problem with Gaussian error distribution (Section A.1), as estimated by the methodology of Section 7.2.

Table 1: Cryptographic parameters for FrodoKEM-640, FrodoKEM-976, FrodoKEM-1344, and their corresponding ephemeral variants. For each set, $\mathsf{len}_\mu = \mathsf{len}_{\mathbf{s}} = \mathsf{len}_{\mathbf{k}} = \mathsf{len}_{\mathbf{pkh}} = \mathsf{len}_{\mathbf{ss}} = \ell$.

	(e)FrodoKEM-640	(e)FrodoKEM-976	(e)FrodoKEM-1344
D	15	16	16
q	32768	65536	65536
n	640	976	1344
$\overline{m} = \overline{n}$	8	8	8
B	2	3	4
$\mathsf{len}_{\mathsf{seed}_{\mathbf{A}}}$	128	128	128
$\mathsf{len}_{\mathbf{z}}$	128	128	128
ℓ	128	192	256
len_χ	16	16	16
χ	$\chi_{\text{Frodo-640}}$	$\chi_{\text{Frodo-976}}$	$\chi_{\text{Frodo-1344}}$
SHAKE	SHAKE128	SHAKE256	SHAKE256

Table 2: Size (in bits) of $\mathsf{len}_{\mathsf{seed}_{\mathbf{SE}}}$ and $\mathsf{len}_{\mathsf{salt}}$.

	FrodoKEM-640	FrodoKEM-976	FrodoKEM-1344
$\mathsf{len}_{\mathsf{seed}_{\mathbf{SE}}}$	256	384	512
$\mathsf{len}_{\mathsf{salt}}$	256	384	512
	eFrodoKEM-640	**eFrodoKEM-976**	**eFrodoKEM-1344**
$\mathsf{len}_{\mathsf{seed}_{\mathbf{SE}}}$	128	192	256
$\mathsf{len}_{\mathsf{salt}}$	0	0	0

Table 5 summarizes the sizes, in bytes, of the different inputs and outputs of FrodoKEM. Note that the secret key sizes include the size of the public key, in order to comply with NIST's API guidelines. Specifically, since NIST's decapsulation API does not include an input for the public key, it needs to be included as part of the secret key.

Table 3: Error distributions.

	σ	Probability of (in multiples of 2^{-16})												Rényi	
		0	±1	±2	±3	±4	±5	±6	±7	±8	±9	±10	±11	±12	order	divergence
$\chi_{\text{Frodo-640}}$	2.8	9288	8720	7216	5264	3384	1918	958	422	164	56	17	4	1	200	0.324×10^{-4}
$\chi_{\text{Frodo-976}}$	2.3	11278	10277	7774	4882	2545	1101	396	118	29	6	1			500	0.140×10^{-4}
$\chi_{\text{Frodo-1344}}$	1.4	18286	14320	6876	2023	364	40	2							1000	0.264×10^{-4}

Table 4: Single-user, single-ciphertext security estimates, following the process outlined in Section 6.2, and detailed for Frodo-640 parameters in Section C.1.1. Numbers under **B** (resp. **C**, **Q**) were obtained using the beyond-core-SVP/C-LSF-Sieve cost model (resp. core-SVP/C-LSF-Sieve, core-SVP/Q-RW-Sieve). IND-CPA numbers are obtained by taking the cheapest corresponding attacks in Section 7.2 and subtracting $\log(\bar{n}+\bar{m}) = 4$ bits lost due to Theorem 3.

	target level	failure rate	IND-CPA sec. B	IND-CPA sec. C	IND-CPA sec. Q	IND-CCA sec. (ROM) B	IND-CCA sec. (ROM) C
(e)FrodoKEM-640	1	$2^{-138.7}$	145	134	119	140	130
(e)FrodoKEM-976	3	$2^{-199.6}$	208	195	173	204	192
(e)FrodoKEM-1344	5	$2^{-252.5}$	262	250	223	258	246

Table 5: Size (in bytes) of inputs and outputs of FrodoKEM and eFrodoKEM.

Scheme	secret key sk	public key pk	ciphertext c	shared secret ss
FrodoKEM-640	19,888	9,616	9,752	16
FrodoKEM-976	31,296	15,632	15,792	24
FrodoKEM-1344	43,088	21,520	21,696	32
eFrodoKEM-640	19,888	9,616	9,720	16
eFrodoKEM-976	31,296	15,632	15,744	24
eFrodoKEM-1344	43,088	21,520	21,632	32

7 Justification of security strength

This section discusses the security of FrodoKEM, which is justified both by security reductions (Section 7.1) and by analysis of the best known cryptanalytic attacks (Section 7.2). A summary of the bit-security estimates based on these two methodologies is shown in Table 4.

7.1 Security reductions

In this section, we provide the main theorems supporting the security of FrodoKEM in the ROM and QROM. Refer to Appendix C for a full overview of the reductions.

7.1.1 IND_{n_c,n_u}-CCA security in the random oracle model

The following theorem says that the $\mathsf{SFO}^{\not\perp\prime}$ transform, which we use to construct FrodoKEM from FrodoPKE, generically yields an IND_{n_c,n_u}-CCA-secure KEM (in the classical random oracle model) from an IND_{n_c,n_u}-CPA-secure public-key encryption scheme, even if the KEM and PKE are parameterized by different distributions, provided that those distributions are sufficiently close in terms of Rényi divergence. We present multi-target security bounds,

parameterized by the number of challenge ciphertexts n_{c}, and the number of users n_{u}. A detailed description of the proof steps is given in Section C.1.

To specialize this result to obtain an ordinary (single-key, single-ciphertext) IND-CCA security bound, one merely sets $n_{\mathsf{c}} = n_{\mathsf{u}} = 1$. In this case, the final additive term in the security bound from (3) is zero, and the overall bound matches what was obtained in [NAB+20].

Theorem 1 ($\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CPA PKE $\implies$ $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CCA KEM in classical ROM, with distribution switch)**.**
Let $\mathsf{PKE}_X = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ *be a* $\delta(n_{\mathsf{u}})$*-correct public-key encryption scheme with message space* $\mathcal{M}$ *that is parameterized by a distribution* X*, and let* s *be an upper bound on the total number of samples drawn from* X *by* KeyGen *and* Enc *combined. Let* G_1*,* G_2 *and* F *be independent random oracles, and let* $\mathsf{KEM}_X^{\not\perp\prime} = \mathsf{SFO}^{\not\perp\prime}[\mathsf{PKE}_X, G_1, G_2, F]$ *be the KEM obtained by applying the* $\mathsf{SFO}^{\not\perp\prime}$ *transform from Definition 11 to* PKE_X*. Let* P, Q *be any discrete distributions. There exists a classical algorithm (a reduction)* $\mathcal{B}$ *against the* $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CPA *security of* PKE_Q*, which uses as a "black box" subroutine any* $\mathcal{A}$ *against the* $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CCA *security of* $\mathsf{KEM}_P^{\not\perp\prime}$ *that makes at most* q_{RO} *oracle queries, for which*

$$\mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-CCA}}_{\mathsf{KEM}_P^{\not\perp\prime}}(\mathcal{A}) \le 2 \cdot \left(\left(\left(\frac{6q_{\mathsf{RO}}}{|\mathcal{M}|} + \frac{1}{2^\lambda} + \frac{n_{\mathsf{c}}}{|\mathcal{M}|} + q_{\mathsf{RO}} \cdot \delta(n_{\mathsf{u}}) + 2 \cdot \mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}\text{-CPA}}_{\mathsf{PKE}_Q}(\mathcal{B}) \right) \cdot \exp(s \cdot \mathrm{D}_\alpha(P\|Q)) \right)^{1-1/\alpha} + \delta(n_{\mathsf{u}}) + \frac{q_{\mathsf{RO}}}{|\mathcal{M}|} + \frac{n_{\mathsf{u}} n_{\mathsf{c}}(n_{\mathsf{c}}-1)}{|\mathcal{M}| \cdot 2^{\mathsf{len}_{\mathsf{salt}}}} \right) \tag{3}$$

for any $\alpha > 1$*, where the Rényi divergence* D_α *is defined in Definition 24, and* $\lambda \ge 512$ *for all* FrodoKEM *parameter sets. The total running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$ *plus the time needed to simulate the random oracles.*

We point out that when $P = Q$ are the same distribution, we have $\exp(s \cdot \mathrm{D}_\alpha(P\|Q)) = 1$ for any $\alpha > 1$ and hence can take α to be arbitrarily large, making the exponent $1 - 1/\alpha$ approach 1 in the limit. This special case is a main theorem from [HHK17], and it relates the IND-CCA security of FrodoKEM to the IND-CPA security of FrodoPKE when they use the same error distribution, e.g., χ_{Frodo}.

The proof of Theorem 1 combines components from three separate works: the modular analysis of the Fujisaki–Okamoto transform by Hofheinz, Hövelmanns and Kiltz [HHK17], the work on tight multi-key and multi-ciphertext security for key encapsulation in [Gla24, GHS25], and the work of Langlois, Stehlé and Steinfeld [LSS14] relating the security of search problems when one distribution is substituted by another via analysis of the Rényi divergence. More specifically, the proof of the theorem proceeds in the following steps:

1. We apply [GHS25, Theorem 23], which shows that the ST transform converts an $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CPA-secure public-key encryption scheme PKE_Q to an $\mathsf{OW}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-PCA-secure public-key encryption scheme with deterministic encryption (in the random oracle model). Note that [GHS25, Theorem 23] is adapted from [HHK17, Theorem 3.2].
2. Next, we apply distribution substitution for the OW-PCA security experiment (which represents a search problem), to switch from distribution Q to P.
3. Finally, we apply [GHS25, Theorem 24], which shows that their modified $\mathsf{U}^{\not\perp}$ transform converts a salted $\mathsf{OW}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-PCA-secure public-key encryption scheme to an $\mathsf{IND}_{n_{\mathsf{c}},n_{\mathsf{u}}}$-CCA-secure KEM (in the random oracle model).

Hofheinz et al. [HHK17] denote the composition of the T and $\mathsf{U}^{\not\perp}$ transforms as the $\mathsf{FO}^{\not\perp}$ transform. As described in Section 4.2, we use a variant of this transform called $\mathsf{SFO}^{\not\perp\prime}$, which differs from $\mathsf{FO}^{\not\perp}$ as follows:

- The T transform is replaced with the ST transform.

- $\mathsf{SFO}^{\not\perp\prime}$ uses a single hash function (with longer output) to compute r and K, whereas $\mathsf{FO}^{\not\perp}$ uses two separate functions, but these are equivalent when the hash functions are modeled as independent random oracles and have appropriate output lengths.
- The $\mathsf{SFO}^{\not\perp\prime}$ computation of r and K also takes the hash $G_1(pk)$ of the public key pk as input, whereas $\mathsf{FO}^{\not\perp}$ does not; this change preserves the relevant theorems (with trivial changes to the proofs), and has the potential to provide stronger multi-key security.

If we apply all the above changes except for the replacement of T by ST, this results in the $\mathsf{FO}^{\not\perp\prime}$ transform [BDK+18, NAB+20]. The $\mathsf{FO}^{\not\perp\prime}$ transform is applied to FrodoPKE to build eFrodoKEM.

7.1.2 IND-CCA security in the quantum random oracle model

Jiang et al. [JZC+18] show that the $\mathsf{FO}^{\not\perp}$ transform yields an IND-CCA-secure KEM from an OW-CPA-secure public-key encryption scheme, in the *quantum* random oracle model. In [GHS25, Theorems 17, 18, 22], this result was extended to show that the $\mathsf{SFO}^{\not\perp}$ transform generically yields an IND_{n_c,n_u}-CCA-secure KEM from a γ-spread IND_{n_c,n_u}-CPA secure PKE in the QROM, and this result extends to the $\mathsf{SFO}^{\not\perp\prime}$ transform. This result is established in two parts:

1. a tight PKE-to-KEM IND_{n_c,n_u}-CPA security reduction, and
2. a well established result about the simulatability of decapsulation oracles for KEMs built using a deterministic encryption scheme.

The second part of the proof of [GHS25, Theorems 17, 18, 25, 27] also introduces a new correctness notion, bounding correctness errors by the advantage of an adversary in an experiment called FFP (find failing plaintext). This advantage is upper bounded in [GHS25, Theorem 29] by $10(q + q_D + 1)^2 \cdot \delta(n_u)$.

Theorem 2 (IND_{n_c,n_u}-CPA PKE $\implies$ IND_{n_c,n_u}-CPA KEM in quantum ROM)**.** *Let* $\mathsf{PKE} = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ *be a* γ*-spread and* $\delta(n_u)$*-correct public-key encryption scheme with message space* $\mathcal{M}$*. Let* G_1*,* G_2 *and* F *be independent random oracles, and let* $\mathsf{KEM}^{\not\perp\prime} = \mathsf{SFO}^{\not\perp\prime}[\mathsf{PKE}, G_1, G_2]$ *be the KEM obtained by applying the* $\mathsf{SFO}^{\not\perp\prime}$ *transform from Definition 11 to* PKE*. Let* $\mathcal{A}$ *be a quantum algorithm against the* IND_{n_c,n_u}-CCA *security of* $\mathsf{KEM}^{\not\perp\prime}$ *that issues at most* q *many queries in total to* G_1 *and* G_2*, with a total depth of* d*. There exists a quantum algorithm (a reduction)* $\mathcal{B}$ *against the* IND_{n_c,n_u}-CPA *security of* PKE*, which uses* $\mathcal{A}$ *as a "black box" subroutine, for which*

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-CPA}}_{\mathsf{KEM}^{\not\perp\prime}}(\mathcal{A}) \le 4\sqrt{d \cdot \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-CPA}}_{\mathsf{PKE}}(\mathcal{B})} + \frac{n_u n_c (n_c - 1)}{|\mathcal{M}| \cdot 2^{\mathsf{len}_{\mathsf{salt}}}} + 4\sqrt{d\left(\frac{6q}{|\mathcal{M}|}\right) + \frac{1}{2^\lambda}} \quad (4)$$

$$+ 10(q + q_D + 1)^2 \cdot \delta(n_u) + 12 q_D (q + 4q_D) \cdot 2^{-\gamma/2} \ , \quad (5)$$

where $\lambda \ge 512$ *for all* FrodoKEM *parameter sets. The total running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$ *plus the time needed to simulate the random oracles.*

Remark 2. Glabush et al. [GHS25, Section 5.2] note that for many real world schemes, the additive loss in advantage due to γ is small enough to be neglected. In [HHM22, Lemma 6], it was shown that FrodoPKE is γ-spread for

$$\gamma = 10240 \ (\mathsf{FrodoPKE\text{-}640}), \quad \gamma = 15616 \ (\mathsf{FrodoPKE\text{-}976}), \quad \gamma = 10752 \ (\mathsf{FrodoPKE\text{-}1344}) \ .$$

7.2 Cryptanalysis

In this section, we provide a renewed cryptanalysis of the LWE instances underlying the FrodoKEM parameter sets. Originally proposed as part of the FrodoKEM submission to the

NIST PQC standardization process in [NAB+20], these LWE instances received a thorough cryptanalysis in the style of ML-KEM [Nat24b, SAB+20]. Such analysis proposes two computational models, following either the *core-SVP* or the *beyond-core-SVP* methodology, to estimate the cost of lattice-reduction attacks on the LWE problem. The specific estimates were then generated using the `pqsec.py` script attached to the third-round submission archive[8] for core-SVP, and using the "leaky-LWE-estimator" script introduced in [DDGR20][9] for beyond-core-SVP.

To revalidate our security claims with a different attack-optimizer codebase and to extend the analysis to account for results published since [NAB+20], we evaluate the cost of lattice attacks using the popular "lattice-estimator" script.[10] Overall, we keep the internals of the estimator unmodified, and only customize the cost models in order to capture different hypothetical attack scenarios. Our cryptanalysis estimation scripts are available at `https://github.com/microsoft/PQCrypto-LWEKE` (folder `estimates`).

When estimating the cost of solving the FrodoKEM LWE instances, the adversary can choose to attack those defined by the public key, or those defined by a ciphertext. They only differ in that the ciphertext contains $\bar{n}$ more LWE samples than the public key, making a public key instance always at least as hard as a ciphertext instance. For this reason, we estimate only the hardness of attacking the ciphertext. Because $\bar{n}$ is a small integer, the difference in hardness is very small.

7.2.1 Estimating the Cost of Lattice Reduction

Relevant lattice-reduction attacks for our analysis are the primal and dual attacks. Both require strong lattice reduction, capable of producing lattice bases of significantly better quality than the ones given as input. State-of-the-art solutions use *block reduction*, where only adjacent subsets of basis vectors, or blocks, are considered during each step. Oracles for solving the SVP or approximate-SVP [LN24] problems are called on orthogonal projections of these blocks, and the short vectors output are then integrated into the original basis.

The general approach just described is behind most strong lattice reduction algorithms, such as BKZ [SE91, SE94], Progressive-BKZ [AWHT16], Self-dual BKZ [MW16], and G6K [ADH+19]. The leading term in the cost (in terms of runtime, memory, or energy) in all of these algorithms is the cost of the (approximate) SVP solver. Assessing the cost of lattice reduction hence reduces to determining the cost of SVP solving, and the number of times the SVP solver must be called before good reduction is achieved, i.e., before a basis of good enough quality is output. The targeted notion of reduction is that of BKZ-β-reducedness [SE91, SE94], which requires that every first vector of a block of rank β in a reduced lattice basis is a shortest nonzero vector in the projected lattice generated by that block.

Given LWE distribution parameters and an attack strategy, identifying optimal attack parameters and their implied cost requires a model for the cost and quality of lattice reduction and SVP solving (and, if lattice sieving is used, a model for the number of short vectors returned by the SVP solver). We proceed to describe two possible methodologies for carrying out such estimates, *core-SVP* and *beyond-core-SVP*.

7.2.2 Core-SVP Methodology

The *core-SVP* methodology [ADPS16] is based on the observation that if a BKZ-β-reduced basis has been achieved, the shortest vector problem must have been solved on blocks of rank β. Because the cost of solving the SVPs is the leading term in the cost of lattice

[8] `https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/submissions/FrodoKEM-Round3.zip`

[9] `https://github.com/lducas/leaky-LWE-Estimator/tree/NIST-round3`

[10] `https://github.com/malb/lattice-estimator`, commit `5ba00f5`.

reduction, the overall cost can be approximated by considering only the smaller cost of a single call to an SVP solver. This is a conservative choice, but it also adds a margin against possible improvements in block reduction that could reduce the number of SVP calls required to obtain a BKZ-β-reduced basis.

The advantage of core-SVP estimates resides in the security margins they provide. For this reason, we make various simplifying assumptions to obtain a likely lower bound for the cost of lattice reduction when attacking FrodoKEM.

- We exclusively consider algorithms in the RAM model, where read/write access to memory is essentially free.
- We consider known asymptotic quantum speedups to classical algorithms, where a quantum computation can be run for an arbitrarily long time, and does not incur any quantum memory costs.
- We consider that operations on rank-β vectors cost β arithmetic operations. While cost models targeting ring-based lattices somewhat ignore this cost factor to hedge against any use of possible rotation symmetry in the ring elements, we consider this to be an inherent overhead when working with unstructured lattices.

These concessions to the adversary suggest the use of the asymptotically fastest available SVP solvers, lattice sieves. In particular, we consider sieves from [BDGL16] using locality-sensitive filtering, and proposed quantum speedups using Grover's algorithm [Laa15a] and quantum random walks [CL21]. Due to our focus on lattice sieving, we make two further considerations when estimating the cost of the primal and dual attacks.

- We assume that the "dimensions-for-free" speedups [Duc18] can be used at all times, meaning that in order to solve an SVP in rank β, it suffices to perform lattice sieving in rank $\beta \cdot \mathrm{dff}_\beta$, where $\mathrm{dff}_\beta = \left(1 - \frac{\log(4/3)}{\log(\beta/(2\pi e))}\right) < 1$ whenever $\beta \geq 23$.
- We assume that lattice sieves in rank r return $2^{0.2075r}$ vectors of norm $\sqrt{4/3} \cdot \lambda_1$ in the lattice, where λ_1 is the minimum distance of the rank-r lattice being sieved, which is relevant when estimating the cost of dual attacks.

With these simplifications, we obtain three cost models for lattice reduction outputting BKZ-β-reduced bases. They correspond directly to three models for the cost of solving SVP in rank β:

- C-LSF-Sieve, where SVP in rank β costs $2^{0.292 \cdot \beta \cdot \mathrm{dff}_\beta + \log_2(\beta \cdot \mathrm{dff}_\beta)}$ by counting the number of arithmetic operations required to solve SVP-β using a classical sieve augmented using locality-sensitive filtering (LSF) [BDGL16].
- Q-Grover-Sieve, where SVP in rank β costs $2^{0.265 \cdot \beta \cdot \mathrm{dff}_\beta + \log_2(\beta \cdot \mathrm{dff}_\beta)}$ by applying Grover's algorithm for quantum search to LSF-based sieving [Laa15a].
- Q-RW-Sieve, where SVP in rank β costs $2^{0.257 \cdot \beta \cdot \mathrm{dff}_\beta + \log_2(\beta \cdot \mathrm{dff}_\beta)}$ by applying quantum random walk (RW) algorithms for search [MNRS07] to LSF-based sieving [CL21].

With these cost models, we estimate the cost of the uSVP and BDD primal attacks, the dual attack described above, as well as the dual-sieve-FFT variant [GJ21, MAT22] that recently received new experimental validation in an update of [CMHST25]. The resulting cost estimates are reported in Table 6, where only estimated lattice reduction costs are shown, meaning that for the dual attacks, the cost of the distinguishing step is not included.

7.2.3 Beyond Core-SVP Methodology

While the core-SVP methodology gives us confidence on the baseline hardness of lattice reduction attacks, it is quite pessimistic about the effective security of lattice-based schemes. Having a large security margin is important since it is not possible to exclude future improvements in cryptanalysis. However, relying only on core-SVP estimates may lead to an unnecessary performance penalty by over-specifying parameters required to

Table 6: Attack costs ($\log_2$) against FrodoKEM's underlying LWE instance in the core-SVP methodology (§ 7.2.2).

SVP model	Primal (uSVP)	Primal (BDD)	Dual	Dual-sieve-FFT
		Frodo-640		
Q-RW-Sieve	123.0	124.0	129.3	128.3
Q-Grover-Sieve	126.5	127.5	133.1	132.1
C-LSF-Sieve	138.5	139.5	145.7	144.4
		Frodo-976		
Q-RW-Sieve	177.0	178.0	184.1	183.3
Q-Grover-Sieve	182.2	183.2	189.5	187.8
C-LSF-Sieve	199.8	200.8	207.9	205.4
		Frodo-1344		
Q-RW-Sieve	231.6	232.6	239.6	227.6
Q-Grover-Sieve	238.5	239.5	246.8	233.3
C-LSF-Sieve	261.8	262.8	270.9	254.8

obtain such a large margin over hypothetical attack improvements. It can also penalize cryptanalysis by not acknowledging improvements in practical attack costs that do not affect their core-SVP cost.

In Round 3 of the NIST PQC standardization process, the designers of ML-KEM introduced a *beyond-core-SVP* methodology [SAB+20], which was also adopted by the FrodoKEM team [NAB+20]. They make various further assumptions on the practical hardness of lattice reduction, such as the difference between the costs of BKZ and progressive BKZ, the number of gates required to implement arithmetic operations, and the possibility that lower-than-expected block sizes still lead to successful attacks.

In this section, we make similar assumptions. They are not a strict subset or superset of the ones made in [SAB+20], with our intention being to propose a complementary look at possible attack overheads. In short, we consider the following:

- Ignoring the number of SVP solver calls leads to a significant underestimate of the cost of lattice reduction, when most algorithms will need to call the solver in the order of thousands of times to reduce the primal and dual lattice bases. We instead consider that BKZ requires $8(d-\beta)$ calls to an SVP solver on rank-β blocks to reduce a d-dimensional lattice. Progressive BKZ instead requires $\sum_{\beta'=60}^{\beta}(d-\beta')$ calls, where we consider SVP calls in rank < 60 to be of negligible cost.
- Lattice sieving is a highly memory-intensive algorithm. Accounting for the extra memory cost requires more than a naïve area-time multiplication, since [BDGL16] optimizes its filters for the RAM model. We adopt the recent analysis of the area-time cost of lattice sieving using two-dimensional (2D) memory architectures by Jaques [Jaq24], which re-calibrates locality-sensitive filtering and recursive sieving to optimize costs. This results in the C-2D-Sieve cost model, where SVP in rank β costs $2^{0.3113\cdot\beta\cdot\mathrm{dff}_\beta+\log_2(\beta\cdot\mathrm{dff}_\beta)}$.
- Due to the significant cost of memory in sieving, we also estimate the cost of SVP solving via parallel pruned enumeration as a sanity check. For a fair comparison with the C-2D-Sieve model, we compute the amount of memory required by C-2D-Sieve, and use Assumption 1 in [Jaq24] together with their case-study of the Cores/Memory ratio for an NVIDIA GeForce RTX 4090 graphics card, to estimate the number of cores available for enumeration on a computer that would alternatively be able to run lattice sieving as the SVP solver. We then take the

Table 7: Attack costs ($\log_2$) against FrodoKEM, beyond-core-SVP methodology (§ 7.2.3).

SVP model	Reduction	uSVP	BDD	Dual-sieve-FFT
		Frodo-640		
C-LSF-Sieve	BKZ	154.7	149.8	155.5
C-2D-Sieve	BKZ	163.6	158.6	164.4
C-Para-Enum	BKZ	201.9	196.1	—
C-LSF-Sieve	PBKZ	154.3	150.5	155.1
C-2D-Sieve	PBKZ	163.0	159.1	163.8
C-Para-Enum	PBKZ	200.8	195.9	—
		Frodo-976		
C-LSF-Sieve	BKZ	218.6	212.6	217.6
C-2D-Sieve	BKZ	231.6	225.4	228.9
C-Para-Enum	BKZ	313.1	305.0	—
C-LSF-Sieve	PBKZ	218.2	213.1	217.2
C-2D-Sieve	PBKZ	231.0	225.8	228.4
C-Para-Enum	PBKZ	311.8	304.3	—
		Frodo-1344		
C-LSF-Sieve	BKZ	282.9	275.4	267.2
C-2D-Sieve	BKZ	300.1	292.3	282.4
C-Para-Enum	BKZ	438.3	426.9	—
C-LSF-Sieve	PBKZ	282.4	276.0	266.8
C-2D-Sieve	PBKZ	299.5	292.8	281.8
C-Para-Enum	PBKZ	436.9	426.2	—

asymptotically cheapest approximate-SVP solver known based on pruned enumeration [ABLR21], and obtain the C-Para-Enum cost model, where SVP solving in rank β costs $2^{0.1250\beta \log_2 \beta - 0.654\beta + 25.84 + \log_2 64} / 2^{0.2075\beta \cdot \mathrm{dff}_\beta + \log_2(\beta \cdot \mathrm{dff}_\beta) + \log_2(\text{Cores/Memory})}$.

- We also include the dual-sieve-FFT attack in our cost estimates. The authors of [CMHST25] recently updated their paper with a version of the dual attack that does not rely on the controversial assumptions made in [MAT22]. The paper additionally provides some experimental results supporting the attack with cost estimates close to those claimed in [MAT22]. Despite the fact that many of the recently proposed improvements are controversial [DP23a, PS24, DP23b], and despite the small amount of positive experimental results, we believe it is prudent to assume that the attack is valid and include its cost estimates.
- It is currently unclear how feasible long-running quantum computations in the style of Grover's search or quantum random walks will be. Limiting circuit depth is believed to incur a sharp loss of the quantum computing advantage [JNRV20], severely limiting the resulting speedups [SHRS17, ANS18] on pruned enumeration [BBTV24]. Similar loss of quantum advantage has also been predicted for Grover and quantum-random-walk-based lattice sieving [AGPS20, DGLM24], due to memory access and correction costs, as well as limitations on circuit depth. We therefore ignore conjectured quantum speedups for lattice sieving and enumeration.

We then proceed to re-estimate the primal uSVP and BDD attacks, using the C-LSF-Sieve, C-2D-Sieve and C-Para-Enum SVP cost models, assuming BKZ and Progressive BKZ lattice reduction. The results can be found in Table 7.

The results in Table 7 confirm the significant gap between the predictable practical cost of FrodoKEM's LWE instances in a memory-aware model such as C-2D-Sieve, and the estimates obtained via the core-SVP analysis reported in Table 6.

7.3 Additional KEM properties

Besides confidentiality (shared-secret indistinguishability), there are several other security properties that have been considered for KEMs.

- *Anonymity*, also known as *key privacy*, meaning that a ciphertext does not reveal anything about which public key was used to create it: Grubbs et al. [GMP22] show that FrodoPKE is ANO-CPA and that eFrodoKEM is ANO-CCA (in both cases, assuming decision-LWE).
- *Robustness*, meaning that a recipient (with a given public key) can be certain they were the intended recipient of a ciphertext: Grubbs et al. [GMP22] show that eFrodoKEM is SCFR-CCA (in the random-oracle model).
- *Binding*: Cremers et al. [CDM24] introduce a hierarchy of security properties capturing the degree to which a KEM's outputs determine (or "bind") other values. The notation X-BIND-P-Q means that value P binds value Q, in one of three attack scenarios: honestly generated key pairs (X=HON), honestly generated key pairs where the secret key is given to the adversary (X=LEAK), and adversarially generated key pairs (X=MAL). They show that eFrodoKEM satisfies the properties LEAK-BIND-K-CT, LEAK-BIND-K-PK, LEAK-BIND-K,CT-PK, and LEAK-BIND-K,PK-CT (in the random-oracle model); but, like all implicitly rejecting KEMs, eFrodoKEM does not provide X-BIND-CT-K or X-BIND-CT-PK for any of X=HON,LEAK,MAL.

8 Implementation and performance analysis

An important feature of FrodoKEM is that it is easy to implement and naturally facilitates writing implementations that are compact and run in constant-time. This latter feature aids to avoid common cryptographic implementation mistakes which can lead to key-extraction based on, for instance, timing differences when executing the code.[11]

Our compact implementation of the FrodoKEM scheme consists of slightly more than 250 lines of plain C code.[12] This same code is used for all three security levels to implement FrodoKEM-640, FrodoKEM-976, and FrodoKEM-1344, with parameters changed by a small number of macros at compile-time. Moreover, most of the code is either shared or reused for our implementation of eFrodoKEM. We remark that the separation in two implementations, one for FrodoKEM and one for eFrodoKEM, is only done to provide a simpler and cleaner codebase supporting each API. In particular, the API for eFrodoKEM has been customized to perform a *single* key generation per encapsulation execution.

Computing on matrices—the basic operation in FrodoKEM—allows for easy scaling to different dimensions n. In addition, FrodoKEM uses a modulus q that is always equal or less than 2^{16}. These two combined aspects allow for the full reuse of the matrix functions for the different security levels by instantiating them with the right parameters at build time. Since the modulus q used is always a power of two, implementing arithmetic modulo q is simple, efficient and easy to do in constant-time in modern computer architectures: for instance, computing modulo 2^{16} comes for free when using 16-bit data-types. Moreover, the dimension values were chosen to be divisible by 16 in order to facilitate vectorization optimizations and to simplify the use of AES128 for the generation of the matrix $\mathbf{A}$.

Also the error sampling is designed to be simple and facilitates code reuse: for any security level, FrodoKEM requires 16 bits per sample, and the tables T_χ corresponding to

[11] Nonetheless, care must be taken to avoid timing leaks. In 2020, Guo, Johansson, and Nilsson [GJN20] demonstrated a key-recovery attack on the reference implementation in the Round 2 submission of FrodoKEM by exploiting branching in the computation of **ss** in FrodoKEM.Decaps. This attack can be avoided by ensuring the implementation reads both $\mathbf{k}'$ and $\mathbf{s}$, compares $\mathbf{B}'\|\mathbf{C}$ and $\mathbf{B}''\|\mathbf{C}'$ in a constant-time way that avoids early termination, and sets $\bar{\mathbf{k}}$ using data-independent evaluation.

[12] Our reference and optimized implementations in C are available at: https://github.com/microsoft/PQCrypto-LWEKE.

the discrete cumulative density functions always consist of values that are less than 2^{15}. Hence, a simple function applying inversion sampling (see Algorithm 1) can be instantiated using precomputed tables T_χ. Moreover, due to the small sizes of these precomputed tables constant-time table lookups, needed to protect against attacks based on timing differences, can be implemented almost for free in terms of effort and performance impact.

All our implementations avoid the use of secret address accesses and secret branches and, hence, are protected against timing and cache attacks at the software level.

Performance analysis on x64 Intel. Table 8 summarizes the results of our performance evaluation using a machine equipped with a 3.2GHz Intel Core i7-8700 (Coffee Lake) processor and running Ubuntu 22.04.2 LTS. Following standard practice, TurboBoost was disabled during the tests. For compilation we used GNU GCC version 15.0.1 with the command `gcc -O3 -march=native`.

For the case of generating the matrix $\mathbf{A}$ using AES128, we present the results when using an AES implementation that exploits the cryptographic extension set AES-NI. The corresponding running times for FrodoKEM-640-AES, FrodoKEM-976-AES and FrodoKEM-1344-AES are 0.67 ms, 1.28 ms and 2.17 ms, respectively, obtained by adding the times for encapsulation and decapsulation. This performance is expected to be typical in static key exchange applications where the cost of key generation is amortized across many key encapsulation executions. For the full KEM, the running times are 0.97 ms, 1.91 ms and 3.22 ms, respectively. These timings roughly match the cost of eFrodoKEM in an ephemeral setting (the overhead is about 1% or less).

Our implementation also includes the optional use of AVX2 intrinsic instructions. In our experiments, we observed that this optimization offers a very small performance improvement compared to the plain C implementation. This illustrates that FrodoKEM's algorithms, which are mainly based on matrix operations, facilitate automatic parallelization using vector instructions. Hence, the compiler is able to achieve close to "optimal" performance with little intervention from the programmer.

We note that the performance of FrodoKEM using AES on Intel platforms greatly depends on AES-NI instructions. For example, when turning off the use of these instructions the computing cost of the optimized implementations suffers a more than 20-fold increase.

Table 8 also outlines the performance figures of our implementation when using SHAKE128 for the generation of $\mathbf{A}$. In this case, we use a 4-way implementation of SHAKE that exploits AVX2 instructions. In our tests, we observed that this approach results in a two-fold speedup when compared to a version using a SHAKE implementation written in plain C.

Comparing the use of AES128 and SHAKE128, FrodoKEM using AES, when implemented with AES-NI instructions, is around 3× faster than FrodoKEM using SHAKE with a vectorized implementation. Nevertheless, this comparative result may change drastically if hardware-accelerated instructions such as AES-NI are not available on the targeted platform, or if support for hardware-accelerated instructions for SHA-3 is added in the future.

Performance analysis on ARM. Table 9 details the performance of our implementations on a device powered by a 1.992GHz 64-bit ARM Cortex-A72 (ARMv8) processor and running Ubuntu 24.04.2 LTS. We provide three options which, again, are determined by the way we generate matrix $\mathbf{A}$. The first option uses OpenSSL's AES engine for implementing and accelerating AES128 with the AES cryptographic extensions available on the targeted ARMv8 processor. This implementation uses OpenSSL version 3.0.13 and was compiled with GNU GCC version 14.2.0. The other two options use plain C implementations of AES and SHAKE for generating $\mathbf{A}$ with AES128 and SHAKE128, respectively. For these cases, we use OpenSSL version 3.0.2 and compiled the implementations with GNU GCC version 13.1.0. For all the options we used the command `gcc -O3 -march=native`. Similar to the case of the x64 Intel platform, the overall performance of FrodoKEM is highly dependent

Table 8: Performance (in thousands of cycles) of FrodoKEM on a 3.2GHz Intel Core i7-8700 (Coffee Lake) processor. For the variants using AES128, results are reported using an AES implementation that exploits AES-NI instructions. For the variants using SHAKE128, results are reported using a 4-way vectorized implementation of SHAKE using AVX2 instructions. Cycle counts are rounded to the nearest 10^3 cycles.

Scheme	Cycles ($\times 10^3$)			
	KeyGen	**Encaps**	**Decaps**	**Total** (E+D)
AES using AES-NI				
FrodoKEM-640-AES	938	1,105	1,044	2,149
FrodoKEM-976-AES	2,017	2,105	1,983	4,088
FrodoKEM-1344-AES	3,353	3,597	3,326	6,923
Vectorized SHAKE using AVX2				
FrodoKEM-640-SHAKE	2,806	2,941	2,877	5,818
FrodoKEM-976-SHAKE	6,026	6,096	5,994	12,090
FrodoKEM-1344-SHAKE	10,725	10,643	10,497	21,140

on the performance of the primitive that is used for the generation of the matrix $\mathbf{A}$. Hence, the best performance in this case is achieved when using an AES implementation that exploits the hardware acceleration provided by the ARMv8 cryptographic extensions. The respective running times for the full KEM are 4.98 ms, 9.99 ms and 17.24 ms for FrodoKEM-640-AES, FrodoKEM-976-AES and FrodoKEM-1344-AES, respectively (as above, these timings only have a negligible overhead in comparison to the cost of eFrodoKEM in an ephemeral setting). On the other hand, SHAKE performs significantly better when there is no support for specialized instructions in the targeted platform: using a plain C version of SHAKE is more than 3× faster than using a plain C version of AES.

8.1 Comparison with other algorithms

In this section, we compare the performance profile of FrodoKEM with two other quantum-safe KEMs that are also expected to be deployed and adopted for real-world applications: CRYSTALS-Kyber [BDK+18], which was recently standardized by NIST as ML-KEM [Nat24b], and Classic McEliece [ABC+24]. As stated before, these two KEMs, alongside FrodoKEM, are currently undergoing standardization by ISO.

Table 10 shows that CRYSTALS-Kyber offers the best performance in terms of both speed and bandwidth. In contrast, FrodoKEM exhibits significantly larger key and ciphertext sizes, as well as substantially slower runtimes, often exceeding CRYSTALS-Kyber's figures by more than an order of magnitude. These performance advantages strongly influenced NIST's choice of CRYSTALS-Kyber for standardization.

However, for security-sensitive applications, it can be argued that Classic McEliece provides a more relevant comparison to FrodoKEM. In this case, the significantly larger public key sizes of Classic McEliece, which exceed FrodoKEM's by more than an order of magnitude, may render it impractical for many use cases. Similarly, the high computational cost of the full Classic McEliece protocol presents additional challenges. For example, the runtime of Classic McEliece at NIST level 5 is approximately 65.5 ms on a server-class processor (compare to FrodoKEM's 3.29 ms). Nonetheless, Classic McEliece offers an advantage in certain static key exchange scenarios in which its substantial key generation cost can be amortized over multiple key encapsulation executions.

Table 9: Performance (in thousands of cycles) of FrodoKEM on a 1.992GHz 64-bit ARM Cortex-A72 (ARMv8) processor. Results are reported for three test cases: (i) using an AES implementation exploiting cryptographic extensions, (ii) using an AES implementation written in plain C, and (iii) using a SHAKE implementation written in plain C. Results have been scaled to cycles using the nominal processor frequency. Cycle counts are rounded to the nearest 10^3 cycles.

Scheme	Cycles ($\times 10^3$)			
	KeyGen	**Encaps**	**Decaps**	**Total** (E+D)
AES using cryptographic extensions				
FrodoKEM-640-AES	3,109	3,402	3,414	6,816
FrodoKEM-976-AES	6,515	6,702	6,678	13,380
FrodoKEM-1344-AES	11,010	11,796	11,527	23,323
Plain C AES				
FrodoKEM-640-AES	47,776	48,007	47,958	95,965
FrodoKEM-976-AES	109,922	110,695	110,387	221,082
FrodoKEM-1344-AES	207,752	209,724	209,164	418,888
Plain C SHAKE				
FrodoKEM-640-SHAKE	11,902	12,197	12,219	24,416
FrodoKEM-976-SHAKE	26,334	26,618	26,713	53,331
FrodoKEM-1344-SHAKE	47,506	48,169	48,505	96,674

9 Acknowledgments

We thank the anonymous reviewers for their valuable feedback. The design of FrodoKEM is the culmination of many years of work by the FrodoKEM team, consisting of (in alphabetical order and excluding the authors of this paper) Erdem Alkim, Joppe Bos, Léo Ducas, Ilya Mironov, Valeria Nikolaenko, and Ananth Raghunathan. We are deeply grateful for their contributions. We also thank Craig Costello for his contributions as co-author of FrodoCCS, and Karen Easterbrook and Brian LaMacchia for their support as co-submitters to the NIST PQC process. Finally, we are very grateful to the large number of researchers and governmental institutions that have studied, implemented and given their support to FrodoKEM.

References

[ABC+24] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece: conservative code-based cryptography, 2017–2024. Specification document available at https://classic.mceliece.org/.

[ABD+23] Erdem Alkim, Joppe W. Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. Annex on FrodoKEM updates, 2023. https://frodokem.org/files/FrodoKEM-annex-20230418.pdf.

Table 10: Performance comparison of FrodoKEM with two other algorithms: CRYSTALS-Kyber and Classic McEliece. The results for FrodoKEM were obtained on a 3.2GHz Intel Core i7-8700 (Coffee Lake) processor, while the results for CRYSTALS-Kyber and Classic McEliece are taken from [BL25] (version `supercop-20241022`), corresponding to a 3.0GHz Intel Core i3-8109U (Coffee Lake) processor. For FrodoKEM we present results for the variant using AES128, and for Classic McEliece we use the fast (f) variants. Cycle counts are rounded to the nearest 10^3 cycles.

Scheme	Sizes (in bytes) pk	ctxt	Cycles ($\times 10^3$) KeyGen	Encaps	Decaps	Total (KG+E+D)
NIST security level 1						
Kyber512	800	768	23	36	28	87
FrodoKEM-640-AES	9,616	9,752	957	1,134	1,059	3,150
`mceliece348864f`	261,120	96	30,381	30	118	30,529
NIST security level 3						
Kyber678	1,184	1,088	39	53	42	134
FrodoKEM-976-AES	15,632	15,792	2,043	2,125	2,042	6,210
`mceliece460896f`	524,160	156	97,899	64	245	98,208
NIST security level 5						
Kyber1024	1,568	1,568	55	75	62	192
FrodoKEM-1344-AES	21,520	21,696	3,432	3,609	3,471	10,512
`mceliece6960119f`	1,047,319	194	195,984	120	287	196,391

[ABD+25] Erdem Alkim, Joppe W. Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM Preliminary Standardization Proposal (submitted to ISO), updated Sept. 2025. https://frodokem.org/files/FrodoKEM_standard_proposal_20250929.pdf.

[ABLR21] Martin R. Albrecht, Shi Bai, Jianwei Li, and Joe Rowell. Lattice reduction with approximate enumeration oracles - practical algorithms and concrete performance. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 732–759, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84245-1_25.

[ACFP15] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. Algebraic algorithms for LWE. *ACM Commun. Comput. Algebra*, 49(2):62, 2015. https://eprint.iacr.org/2014/1018. doi:10.1145/2815111.2815158.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, Berlin, Heidelberg, August 2009. doi:10.1007/978-3-642-03356-8_35.

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th Annual ACM Symposium on Theory of Computing*, pages 284–293. ACM Press, May 1997. doi:10.1145/258533.258604.

[ADH+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 Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 717–746. Springer, Cham, May 2019. doi:10.1007/978-3-030-17656-3_25.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016: 25th USENIX Security Symposium*, pages 327–343. USENIX Association, August 2016.

[AFFP14] Martin R. Albrecht, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Lazy modulus switching for the BKW algorithm on LWE. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 429–445. Springer, Berlin, Heidelberg, March 2014. doi:10.1007/978-3-642-54631-0_25.

[AFG14] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-SVP. In Hyang-Sook Lee and Dong-Guk Han, editors, *ICISC 13: 16th International Conference on Information Security and Cryptology*, volume 8565 of *Lecture Notes in Computer Science*, pages 293–310. Springer, Cham, November 2014. doi:10.1007/978-3-319-12160-4_18.

[AGPS20] Martin R. Albrecht, Vlad Gheorghiu, Eamonn W. Postlethwaite, and John M. Schanck. Estimating quantum speedups for lattice sieves. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 583–613. Springer, Cham, December 2020. doi:10.1007/978-3-030-64834-3_20.

[AGVW17] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving uSVP and applications to LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 297–322. Springer, Cham, December 2017. https://eprint.iacr.org/2017/815. doi:10.1007/978-3-319-70694-8_11.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, May 1996. doi:10.1145/237814.237838.

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 103–129. Springer, Cham, April / May 2017. doi:10.1007/978-3-319-56614-6_4.

[ANS18] Yoshinori Aono, Phong Q. Nguyen, and Yixin Shen. Quantum lattice enumeration and tweaking discrete pruning. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 405–434. Springer, Cham, December 2018. doi:10.1007/978-3-030-03326-2_14.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9(3):169–203, Nov 2015. doi:10.1515/jmc-2015-0016.

[AR05] Dorit Aharonov and Oded Regev. Lattice problems in NP ∩ coNP. *Journal of the ACM*, 52(5):749–765, 2005. Preliminary version in FOCS 2004. doi:10.1145/1089023.1089025.

[AWHT16] Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 789–819. Springer, Berlin, Heidelberg, May 2016. doi:10.1007/978-3-662-49890-3_30.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, Berlin, Heidelberg, May 2000. doi:10.1007/3-540-45539-6_18.

[BBTV24] Nina Bindel, Xavier Bonnetain, Marcel Tiepelt, and Fernando Virdia. Quantum lattice enumeration in limited depth. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VI*, volume 14925 of *Lecture Notes in Computer Science*, pages 72–106. Springer, Cham, August 2024. doi:10.1007/978-3-031-68391-6_3.

[BCD+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 Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1006–1018. ACM Press, October 2016. doi:10.1145/2976749.2978425.

[BCK96] M. Bellare, R. Canetti, and H. Krawczyk. Pseudorandom functions revisited: the cascade construction and its concrete security. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 514–523, 1996. doi:10.1109/SFCS.1996.548510.

[BCNS15] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE Computer Society Press, May 2015. doi:10.1109/SP.2015.40.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 10–24. ACM-SIAM, January 2016. doi:10.1137/1.9781611974331.ch2.

[BDK+18] Joppe 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 (EuroS&P)*, pages 353–367, April 2018. doi:10.1109/EuroSP.2018.00032.

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

[BFKL94] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, Berlin, Heidelberg, August 1994. doi:10.1007/3-540-48329-2_24.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556. Springer, Berlin, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_30.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325. Association for Computing Machinery, January 2012. doi:10.1145/2090236.2090262.

[BL25] Daniel J. Bernstein and Tanja Lange. SUPERCOP benchmarking results, accessed on January 12, 2025. Available at https://bench.cr.yp.to/supercop.html.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 575–584. ACM Press, June 2013. doi:10.1145/2488608.2488680.

[BNvdP17] Joppe W. Bos, Michael Naehrig, and Joop van de Pol. Sieving for shortest vectors in ideal lattices: a practical perspective. *Int. J. of Applied Cryptography*, 3(4):313–329, January 2017. doi:10.1504/IJACT.2017.10010312.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106. IEEE Computer Society Press, October 2011. doi:10.1109/FOCS.2011.12.

[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 Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024: 31st Conference on Computer and Communications Security*, pages 1046–1060. ACM Press, October 2024. doi:10.1145/3658644.3670283.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, Berlin, Heidelberg, August 2005. doi:10.1007/11535218_26.

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 559–585. Springer, Berlin, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_20.

[CDW17] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-SVP. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 324–348. Springer, Cham, April / May 2017. doi:10.1007/978-3-319-56620-7_12.

[CGS14] Peter Campbell, Michael Groves, and Dan Shepherd. Soliloquy: a cautionary tale. ETSI 2nd Quantum-Safe Crypto Workshop, 2014. http://docbox.etsi.org/Workshop/2014/201410_CRYPTO/S07_Systems_and_Attacks/S07_Groves_Annex.pdf.

[Che13] Yuanmi Chen. *Lattice reduction and concrete security of fully homomorphic encryption*. PhD thesis, Université Paris Diderot, 2013.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, October 2012. doi:10.1007/s00145-011-9105-2.

[CIV16] Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Provably weak instances of ring-LWE revisited. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 147–167. Springer, Berlin, Heidelberg, May 2016. doi:10.1007/978-3-662-49890-3_6.

[CL21] André Chailloux and Johanna Loyer. Lattice sieving via quantum random walks. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 63–91. Springer, Cham, December 2021. doi:10.1007/978-3-030-92068-5_3.

[CLS16] Hao Chen, Kristin Lauter, and Katherine E. Stange. Vulnerable galois RLWE families and improved attacks. Cryptology ePrint Archive, Report 2016/193, 2016. URL: https://eprint.iacr.org/2016/193.

[CLS17] Hao Chen, Kristin Lauter, and Katherine E. Stange. Attacks on the search RLWE problem with small errors. *SIAM J. Appl. Algebra Geom.*, 1(1):665–682, 2017. https://eprint.iacr.org/2015/971. doi:10.1137/16M1096566.

[CMHST25] 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 Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part I*, volume 16000 of *Lecture Notes in Computer Science*, pages 444–476. Springer, 2025. doi:10.1007/978-3-032-01855-7_15.

[CN97] Jin-yi Cai and Ajay Nerurkar. An improved worst-case to average-case connection for lattice problems. In *38th Annual Symposium on Foundations of Computer Science*, pages 468–477. IEEE Computer Society Press, October 1997. doi:10.1109/SFCS.1997.646135.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Berlin, Heidelberg, December 2011. doi:10.1007/978-3-642-25385-0_1.

[DDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020*, volume 12171 of *Lecture Notes in Computer Science*, pages 329–358. Springer, 2020. https://eprint.iacr.org/2020/292.pdf. doi:10.1007/978-3-030-56880-1_12.

[DGLM24] 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, Paper 2024/1692, 2024. https://eprint.iacr.org/2024/1692.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.

[DHK+21] Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, and Gregor Seiler. Faster lattice-based KEMs via a generic fujisaki-okamoto transform using prefix hashing. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021: 28th Conference on Computer and Communications Security*, pages 2722–2737. ACM Press, November 2021. doi:10.1145/3460120.3484819.

[DM13] Nico Döttling and Jörn Müller-Quade. Lossy codes and a new variant of the learning-with-errors problem. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 18–34. Springer, Berlin, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_2.

[DP23a] Léo Ducas and Ludo N. Pulles. Does the dual-sieve attack on learning with errors even work? In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 37–69. Springer, Cham, August 2023. doi:10.1007/978-3-031-38548-3_2.

[DP23b] Léo Ducas and Ludo N. Pulles. Accurate score prediction for dual-sieve attacks. Cryptology ePrint Archive, Paper 2023/1850, 2023. URL: https://eprint.iacr.org/2023/1850.

[DRS14] Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. On the closest vector problem with a distance guarantee. In *IEEE Conference on Computational Complexity*, pages 98–109, 2014. doi:10.1109/CCC.2014.18.

[Duc18] Léo Ducas. Shortest vector from lattice sieving: A few dimensions for free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 125–145. Springer, Cham, April / May 2018. doi:10.1007/978-3-319-78381-9_5.

[Dwo15] Morris J. Dworkin. SHA-3 standard: Permutation-based hash and extendable-output functions. Federal Information Processing Standards (FIPS) 202, National Institute of Standards and Technology, August 2015.

[DXL12] Jintai Ding, Xiang Xie, and Xiaodong Lin. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688, 2012. URL: https://eprint.iacr.org/2012/688.

[ELOS15] Yara Elias, Kristin E. Lauter, Ekin Ozman, and Katherine E. Stange. Provably weak instances of ring-LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 63–92. Springer, Berlin, Heidelberg, August 2015. doi:10.1007/978-3-662-47989-6_4.

[Fed24] Federal Office for Information Security (BSI). Cryptographic Mechanisms: Recommendations and Key Lengths, BSI TR-02102-1, Version: 2024-1, 2024. Available at https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile&v=7.

[FO99a] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *PKC'99: 2nd International Workshop on Theory and Practice in Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer, Berlin, Heidelberg, March 1999. doi:10.1007/3-540-49162-7_5.

[FO99b] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, Berlin, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_34.

[Gen22] General Intelligence and Security Service (AIVD). Prepare for the threat of quantum computers, 2022. Available at https://english.aivd.nl/publications/publications/2022/01/18/prepare-for-the-threat-of-quantumcomputers.

[Gen24] General Intelligence and Security Service (AIVD) and Centrum Wiskunde & Informatica (CWI) and Netherlands Organization for Applied Scientific Research (TNO). The PQC Migration Handbook: Guidelines for Migrating to Post-Quantum Cryptography (second edition), 2024. Available at https://publications.tno.nl/publication/34643386/fXcPVHsX/TNO-2024-pqc-en.pdf.

[GGH96] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. Cryptology ePrint Archive, Report 1996/009, 1996. URL: https://eprint.iacr.org/1996/009.

[GHS25] Lewis Glabush, Kathrin Hövelmanns, and Douglas Stebila. Tight multi-challenge security reductions for key encapsulation mechanisms. Cryptology ePrint Archive, Paper 2025/343, 2025. URL: https://eprint.iacr.org/2025/343.

[GJ21] Qian Guo and Thomas Johansson. Faster dual lattice attacks for solving LWE with applications to CRYSTALS. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 33–62. Springer, Cham, December 2021. 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 Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 359–386. Springer, Cham, August 2020. doi:10.1007/978-3-030-56880-1_13.

[Gla24] Lewis Glabush. Tight multi-target security for key encapsulation mechanisms, 2024. https://uwspace.uwaterloo.ca/items/8871c4b1-c2c6-4a42-9485-1466f1772985.

[GMP22] Paul Grubbs, Varun Maram, and Kenneth G. Paterson. Anonymous, robust post-quantum public key encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 402–432. Springer, Cham, May / June 2022. doi:10.1007/978-3-031-07082-2_15.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, May 2008. doi:10.1145/1374376.1374407.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, Berlin, Heidelberg, August 2013. doi:10.1007/978-3-642-40041-4_5.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 503–523. Springer, Berlin, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_25.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371. Springer, Cham, November 2017. 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 Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 414–443. Springer, Cham, December 2022. doi:10.1007/978-3-031-22972-5_15.

[Höv21] Kathrin Hövelmanns. *Generic constructions of quantum-resistant cryptosystems*. doctoralthesis, Ruhr-Universität Bochum, Universitätsbibliothek, 2021. doi:10.13154/294-7758.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, June 1998. doi:10.1007/BFB0054868.

[IKMT14] Tsukasa Ishiguro, Shinsaku Kiyomoto, Yutaka Miyake, and Tsuyoshi Takagi. Parallel gauss sieve algorithm: Solving the SVP challenge over a 128-dimensional ideal lattice. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 411–428. Springer, Berlin, Heidelberg, March 2014. doi:10.1007/978-3-642-54631-0_24.

[Int24] International Organization for Standardization (ISO). ISO/IEC 18033-2:2006/DAmd 2, Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers, 2024. Available at https://www.iso.org/standard/86890.html.

[Jaq24] Samuel Jaques. Memory adds no cost to lattice sieving for computers in 3 or more spatial dimensions. *IACR Communications in Cryptology (CiC)*, 1(3):6, 2024. doi:10.62056/ay4fbn2hd.

[JNRV20] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. Implementing grover oracles for quantum key search on AES and LowMC. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 280–310. Springer, Cham, May 2020. doi:10.1007/978-3-030-45724-2_10.

[JZC+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 Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 96–125. Springer, Cham, August 2018. doi:10.1007/978-3-319-96878-0_4.

[KBF+15] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Yu Chen, Z. Chen, B. Chiaro, A. Dunsworth, I.-C. Hoi, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner, A. N. Cleland, and John M. Martinis. State preservation by repetitive error detection in a superconducting quantum circuit. *Nature*, 519(7541):66–69, March 2015. doi:10.1038/nature14270.

[KF15] Paul Kirchner and Pierre-Alain Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 43–62. Springer, Berlin, Heidelberg, August 2015. doi:10.1007/978-3-662-47989-6_3.

[KF17] Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched NTRU parameters. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 3–26. Springer, Cham, April / May 2017. doi:10.1007/978-3-319-56620-7_1.

[KTX07] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Multi-bit cryptosystems based on lattice problems. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 315–329. Springer, Berlin, Heidelberg, April 2007. doi:10.1007/978-3-540-71677-8_21.

[Laa15a] Thijs Laarhoven. *Search problems in cryptography*. PhD thesis, Eindhoven University of Technology, 2015.

[Laa15b] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 3–22. Springer, Berlin, Heidelberg, August 2015. doi:10.1007/978-3-662-47989-6_1.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982.

[LLM06] Yi-Kai Liu, Vadim Lyubashevsky, and Daniele Micciancio. On bounded distance decoding for general lattices. In Josep Díaz, Klaus Jansen, José D. P. Rolim, and Uri Zwick, editors, *APPROX-RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pages 450–461. Springer, 2006. doi:10.1007/11830924_41.

[LN13] Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In Ed Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 293–309. Springer, Berlin, Heidelberg, February / March 2013. doi:10.1007/978-3-642-36095-4_19.

[LN24] Jianwei Li and Phong Q. Nguyen. A complete analysis of the bkz lattice reduction algorithm. *J. Cryptol.*, 38(1), December 2024. doi:10.1007/s00145-024-09527-0.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, Berlin, Heidelberg, February 2011. doi:10.1007/978-3-642-19074-2_21.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM*, 60(6):43:1–43:35, November 2013. Preliminary version in Eurocrypt 2010. doi:10.1145/2535925.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015. doi:10.1007/S10623-014-9938-4.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 239–256. Springer, Berlin, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_14.

[MAT22] MATZOV. Report on the security of LWE: Improved dual lattice attack. Technical report, Israel Defense Forces, 2022.

[Mic02] Daniele Micciancio. Improved cryptographic hash functions with worst-case/average-case connection. In *34th Annual ACM Symposium on Theory of Computing*, pages 609–618. ACM Press, May 2002. doi:10.1145/509907.509995.

[Mic10] Daniele Micciancio. Cryptographic functions from worst-case complexity assumptions. In Phong Q. Nguyen and Brigitte Vallée, editors, *The LLL Algorithm - Survey and Applications*, Information Security and Cryptography, pages 427–452. Springer, 2010. doi:10.1007/978-3-642-02295-1_13.

[MNRS07] Frédéric Magniez, Ashwin Nayak, Jeremie Roland, and Miklos Santha. Search via quantum walk. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing*, pages 575–584. ACM Press, June 2007. doi:10.1145/1250790.1250874.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 21–39. Springer, Berlin, Heidelberg, August 2013. doi:10.1007/978-3-642-40041-4_2.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004. doi:10.1137/S0097539705447360.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post Quantum Cryptography*, pages 147–191. Springer, 2009. doi:10.1007/978-3-540-88702-7_5.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, Berlin, Heidelberg, February 2004. doi:10.1007/978-3-540-24638-1_2.

[MW16] Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 820–849. Springer, Berlin, Heidelberg, May 2016. doi:10.1007/978-3-662-49890-3_31.

[NAB+17] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

[NAB+19] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[NAB+20] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[Nat17] National Institute of Standards and Technology (NIST). Post-quantum cryptography standardization project, 2017. Call for proposals: https://frodokem.org/files/FrodoKEM-standard_proposal-20230314.pdf. URL: http://csrc.nist.gov/groups/ST/post-quantum-crypto/.

[Nat23] National Cybersecurity Agency of France (ANSSI). ANSSI views on the Post-Quantum Cryptography transition (2023 follow up), 2023. Available at https://cyber.gouv.fr/en/publications/follow-position-paper-post-quantum-cryptography.

[Nat24a] National Institute of Standards and Technology (NIST). Module-Lattice-Based Digital Signature Standard (FIPS 204), 2024. Available at https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf.

[Nat24b] National Institute of Standards and Technology (NIST). Module-Lattice-Based Key-Encapsulation Mechanism Standard (FIPS 203), 2024. Available at https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf.

[OP01] Tatsuaki Okamoto and David Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In David Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 159–175. Springer, Berlin, Heidelberg, April 2001. doi:10.1007/3-540-45353-9_13.

[Pei09a] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342. ACM Press, May / June 2009. doi:10.1145/1536414.1536461.

[Pei09b] Chris Peikert. Some recent progress in lattice-based cryptography. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, page 72. Springer, Berlin, Heidelberg, March 2009. Invited talk. Slides available at http://web.eecs.umich.edu/~cpeikert/pubs/slides-tcc09.pdf.

[Pei10] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97. Springer, Berlin, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_5.

[Pei14] Chris Peikert. Lattice cryptography for the internet. In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014*, pages 197–219. Springer, Cham, October 2014. doi:10.1007/978-3-319-11659-4_12.

[Pei16a] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016. doi:10.1561/0400000074.

[Pei16b] Chris Peikert. How (not) to instantiate ring-LWE. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16: 10th International Conference on Security in Communication Networks*, volume 9841 of *Lecture Notes in Computer Science*, pages 411–430. Springer, Cham, August / September 2016. doi:10.1007/978-3-319-44618-9_22.

[Per21] Ray Perlner. Multi-ciphertext attacks, August 2021. Private communication.

[PHS19] Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. Approx-SVP in ideal lattices with pre-processing. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 685–716. Springer, Cham, May 2019. doi:10.1007/978-3-030-17656-3_24.

[PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 461–473. ACM Press, June 2017. doi:10.1145/3055399.3055489.

[PS24] Amaury Pouly and Yixin Shen. Provable dual attacks on learning with errors. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part VII*, volume 14657 of *Lecture Notes in Computer Science*, pages 256–285. Springer, Cham, May 2024. doi:10.1007/978-3-031-58754-2_10.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, Berlin, Heidelberg, August 2008. doi:10.1007/978-3-540-85174-5_31.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196. ACM Press, May 2008. doi:10.1145/1374376.1374406.

[Reg04] Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004. Preliminary version in STOC 2003. doi:10.1145/1039488.1039490.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34, 2009. Preliminary version in STOC 2005. doi:10.1145/1568318.1568324.

[Reg10] Oded Regev. The learning with errors problem (invited survey). In *IEEE Conference on Computational Complexity*, pages 191–204, 2010. doi:10.1109/CCC.2010.26.

[SAB+20] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987. doi:10.1016/0304-3975(87)90064-8.

[Sch13] Michael Schneider. Sieving for shortest vectors in ideal lattices. In Amr Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien, editors, *AFRICACRYPT 13: 6th International Conference on Cryptology in Africa*, volume 7918 of *Lecture Notes in Computer Science*, pages 375–391. Springer, Berlin, Heidelberg, June 2013. doi:10.1007/978-3-642-38553-7_22.

[SE91] Claus-Peter Schnorr and M Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In Lothar Budach, editor, *FCT*, volume 529 of *Lecture Notes in Computer Science*, pages 68–85. Springer, 1991. doi:10.1007/3-540-54458-5_51.

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66:181–199, 1994. doi:10.1007/BF01581144.

[SHRS17] John M. Schanck, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. NTRU-HRSS-KEM. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 192–216. Springer, Berlin, Heidelberg, October / November 2016. doi:10.1007/978-3-662-53644-5_8.

[YN05] J. Q. You and Franco Nori. Superconducting circuits and quantum information. *Physics Today*, 58(11):42–47, November 2005. URL: http://dx.doi.org/10.1063/1.2155757, doi:10.1063/1.2155757.

A LWE and lattices

Here, we describe the mathematical background on learning with errors, Gaussians and lattices that is necessary to analyze the security of FrodoPKE and FrodoKEM.

A.1 Learning with errors

The security of our proposed PKE and KEM relies on the hardness of the *Learning With Errors* (LWE) problem, a generalization of the classic Learning Parities with Noise problem (see, e.g., [BFKL94]) first defined by Regev [Reg09]. This section defines the LWE probability distributions and computational problems.

Definition 12 (LWE distribution)**.** Let n, q be positive integers, and let χ be a distribution over $\mathbb{Z}$. For an $\mathbf{s} \in \mathbb{Z}_q^n$, the *LWE distribution* $A_{\mathbf{s},\chi}$ is the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained

by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random and an integer error $e \in \mathbb{Z}$ from χ, and outputting the pair $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

There are two main kinds of computational LWE problem: *search*, which is to recover the secret $\mathbf{s} \in \mathbb{Z}_q^n$ given a certain number of samples drawn from the LWE distribution $A_{\mathbf{s},\chi}$; and *decision*, which is to distinguish a certain number of samples drawn from the LWE distribution from uniformly random samples. For both variants, one often considers two distributions of the secret $\mathbf{s} \in \mathbb{Z}_q^n$: the uniform distribution, and the distribution $\chi^n \bmod q$ where each coordinate is drawn from the error distribution χ and reduced modulo q. The latter is often called the "normal form" of LWE.

Definition 13 (LWE Search Problem)**.** Let n, m, q be positive integers, and let χ be a distribution over $\mathbb{Z}$. The *uniform-secret* (respectively, *normal-form*) learning with errors *search* problem with parameters (n, m, q, χ), denoted by $\mathsf{SLWE}_{n,m,q,\chi}$ (respectively, $\mathsf{nf\text{-}SLWE}_{n,m,q,\chi}$), is as follows: given m samples from the LWE distribution $A_{\mathbf{s},\chi}$ for uniformly random $\mathbf{s}$ (resp, $\mathbf{s} \leftarrow\!\!\$\ \chi^n \bmod q$), find $\mathbf{s}$. More formally, for an adversary $\mathcal{A}$, define (for the uniform-secret case)

$$\mathrm{Adv}^{\mathsf{slwe}}_{n,m,q,\chi}(\mathcal{A}) = \Pr\left[\mathcal{A}((\mathbf{a}_i, b_i))_{i=1,\ldots,m}) \Rightarrow \mathbf{s} : \mathbf{s} \leftarrow\!\!\$\ U(\mathbb{Z}_q^n), (\mathbf{a}_i, b_i) \leftarrow\!\!\$\ A_{\mathbf{s},\chi}, i = 1, \ldots, m\right] .$$

Similarly, define (for the normal-form case) $\mathrm{Adv}^{\mathsf{nf\text{-}slwe}}_{n,m,q,\chi}(\mathcal{A})$, where $\mathbf{s} \leftarrow\!\!\$\ \chi^n \bmod q$ instead of $\mathbf{s} \leftarrow\!\!\$\ U(\mathbb{Z}_q^n)$.

Definition 14 (LWE Decision Problem)**.** Let n, m, q be positive integers, and let χ be a distribution over $\mathbb{Z}$. The *uniform-secret* (respectively, *normal-form*) *learning with errors decision problem* with parameters (n, m, q, χ), denoted $\mathsf{DLWE}_{n,m,q,\chi}$ (respectively, $\mathsf{nf\text{-}DLWE}_{n,m,q,\chi}$), is as follows: distinguish m samples drawn from the LWE distribution $A_{\mathbf{s},\chi}$ from m samples drawn from the uniform distribution $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$. More formally, for an adversary $\mathcal{A}$, define (for the uniform-secret case)

$$\begin{aligned}\mathrm{Adv}^{\mathsf{dlwe}}_{n,m,q,\chi}(\mathcal{A}) = \Big| &\Pr\left[\mathcal{A}((\mathbf{a}_i, b_i)_{i=1,\ldots,m}) \Rightarrow 1 : \mathbf{s} \leftarrow\!\!\$\ U(\mathbb{Z}_q^n), (\mathbf{a}_i, b_i) \leftarrow\!\!\$\ A_{\mathbf{s},\chi}, i = 1, \ldots, m\right] \\ &- \Pr\left[\mathcal{A}((\mathbf{a}_i, b_i)_{i=1,\ldots,m}) \Rightarrow 1 : (\mathbf{a}_i, b_i) \leftarrow\!\!\$\ U(\mathbb{Z}_q^n \times \mathbb{Z}_q), i = 1, \ldots, m\right]\Big| .\end{aligned}$$

Similarly, define (for the normal-form case) $\mathrm{Adv}^{\mathsf{nf\text{-}dlwe}}_{n,m,q,\chi}(\mathcal{A})$, where $\mathbf{s} \leftarrow\!\!\$\ \chi^n \bmod q$ instead of $\mathbf{s} \leftarrow\!\!\$\ U(\mathbb{Z}_q^n)$.

For all of the above problems, when $\chi = \Psi_{\alpha q}$ is the continuous Gaussian of parameter αq, rounded to the nearest integer (see Definition 16 below), we often replace the subscript χ by α.

A.2 Gaussians

For any real $s > 0$, the (one-dimensional) *Gaussian function* with parameter (or width) s is the function $\rho_s \colon \mathbb{R} \to \mathbb{R}^+$, defined as $\rho_s(\mathbf{x}) := \exp(-\pi \|\mathbf{x}\|^2 / s^2)$.

Definition 15 (Gaussian distribution)**.** For any real $s > 0$, the (one-dimensional) *Gaussian distribution* with parameter (or width) s, denoted D_s, is the distribution over $\mathbb{R}$ having probability density function $D_s(x) = \rho_s(x)/s$.

Note that D_s has standard deviation $\sigma = s/\sqrt{2\pi}$.

Definition 16 (Rounded Gaussian distribution)**.** For any real $s > 0$, the *rounded Gaussian distribution* with parameter (or width) s, denoted Ψ_s, is the distribution over $\mathbb{Z}$ obtained by rounding a sample from D_s to the nearest integer:

$$\Psi_s(x) = \int_{\{z \colon \lfloor z \rceil = x\}} D_s(z)\, dz .$$

A.3 Lattices

Here we recall some background on lattices that will be used when relating LWE to lattice problems.

Definition 17 (Lattice)**.** A (full-rank) *n-dimensional lattice* $\mathcal{L}$ is a discrete additive subset of $\mathbb{R}^n$ for which $\mathrm{span}_{\mathbb{R}}(\mathcal{L}) = \mathbb{R}^n$. Any such lattice can be generated by a (non-unique) *basis* $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^n$ of linearly independent vectors, as $\mathcal{L} = \mathcal{L}(\mathbf{B}) := \mathbf{B} \cdot \mathbb{Z}^n = \left\{\sum_{i=1}^n z_i \cdot \mathbf{b}_i : z_i \in \mathbb{Z}\right\}$. The *volume*, or *determinant*, of $\mathcal{L}$ is defined as $\mathrm{vol}(\mathcal{L}) := |\det(\mathbf{B})|$. An *integer lattice* is a lattice that is a subset of $\mathbb{Z}^n$. For an integer q, a *q-ary lattice* is an integer lattice that contains $q\mathbb{Z}^n$.

Definition 18 (Minimum distance)**.** For a lattice $\mathcal{L}$, its *minimum distance* is the length (in the Euclidean norm) of a shortest non-zero lattice vector: $\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{v}\|$. More generally, its *ith successive minimum* $\lambda_i(\mathcal{L})$ is the smallest real $r > 0$ such that $\mathcal{L}$ has i linearly independent vectors of length at most r.

Definition 19 (Discrete Gaussian)**.** For a lattice $\mathcal{L} \subset \mathbb{R}^n$, the *discrete Gaussian distribution* over $\mathcal{L}$ with parameter s, denoted $D_{\mathcal{L},s}$, is defined as $D_s(\mathbf{x}) = \rho_s(\mathbf{x})/\rho_s(\mathcal{L})$ for $\mathbf{x} \in \mathcal{L}$ (and $D_s(\mathbf{x}) = 0$ otherwise), where $\rho_s(\mathcal{L}) = \sum_{\mathbf{v} \in \mathcal{L}} \rho_s(\mathbf{v})$ is a normalization factor.

We now recall various computational problems on lattices. We stress that these are *worst-case* problems, i.e., to solve such a problem an algorithm must succeed on *every* input (and not just on a randomly chosen input from some probability distribution). The following two problems are parameterized by an *approximation factor* $\gamma = \gamma(n)$, which is a function of the lattice dimension n.

Definition 20 (Decisional approximate shortest vector problem (GapSVP_γ))**.** Given a basis $\mathbf{B}$ of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, where $\lambda_1(\mathcal{L}) \leq 1$ or $\lambda_1(\mathcal{L}) > \gamma(n)$, determine which is the case.

Definition 21 (Approximate shortest independent vectors problem (SIVP_γ))**.** Given a basis $\mathbf{B}$ of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, output a set $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subset \mathcal{L}$ of n linearly independent lattice vectors where $\|\mathbf{v}_i\| \leq \gamma(n) \cdot \lambda_n(\mathcal{L})$ for all i.

The following problem is parameterized by a function φ from lattices to positive real numbers.

Definition 22 (Discrete Gaussian Sampling (DGS_φ))**.** Given a basis $\mathbf{B}$ of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ and a real number $s \geq \varphi(\mathcal{L})$, output a sample from the discrete Gaussian distribution $D_{\mathcal{L},s}$.

B Additional algorithms

This section details the algorithms for matrix encoding and packing required by FrodoPKE and FrodoKEM.

Matrix encoding of bit strings. This subsection describes how bit strings are encoded as mod-q integer matrices. Recall that $2^B \leq q$. The encoding function $\mathrm{ec}(\cdot)$ encodes an integer $0 \leq k < 2^B$ as an element in $\mathbb{Z}_q$ by multiplying it by $q/2^B = 2^{D-B}$: $\mathrm{ec}(k) := k \cdot q/2^B$. This encoding function can be found in early works on LWE-based encryption, for example [KTX07, PW08, PVW08]. Using this function, the function Frodo.Encode encodes bit strings of length $\ell = B \cdot \overline{m} \cdot \overline{n}$ as $\overline{m}$-by-$\overline{n}$-matrices with entries in $\mathbb{Z}_q$ by applying $\mathrm{ec}(\cdot)$ to B-bit sub-strings sequentially and filling the matrix row by row entry-wise. Each B-bit sub-string is interpreted as an integer $0 \leq k < 2^B$ and then encoded by $\mathrm{ec}(k)$, which means

that B-bit values are placed into the B most significant bits of the corresponding entry modulo q.

The corresponding decoding function Frodo.Decode decodes the $\overline{m}$-by-$\overline{n}$ matrix $\mathbf{K}$ into a bit string of length $\ell = B \cdot \overline{m} \cdot \overline{n}$. It extracts B bits from each entry by applying the function $\mathrm{dc}(\cdot)$: $\mathrm{dc}(c) = \lfloor c \cdot 2^B/q \rceil \bmod 2^B$. That is, the $\mathbb{Z}_q$-entry is interpreted as an integer, then divided by $q/2^B$ and rounded. This amounts to rounding to the B most significant bits of each entry. With these definitions, it is the case that $\mathrm{dc}(\mathrm{ec}(k)) = k$ for all $0 \leq k < 2^B$.

Packing matrices modulo q. This subsection specifies packing and unpacking algorithms to transform matrices with entries in $\mathbb{Z}_q$ to bit strings and vice versa. The algorithm Frodo.Pack packs a matrix into a bit string by simply concatenating the D-bit matrix coefficients, as shown in Algorithm 11. Note that in the software implementation, the resulting bit string is stored as a byte array, padding with zero bits to make the length a multiple of 8. The reverse operation Frodo.Unpack is shown in Algorithm 12.

Algorithm 11 Frodo.Pack

Input: Matrix $\mathbf{C} \in \mathbb{Z}_q^{n_1 \times n_2}$.
Output: Bit string $\mathbf{b} \in \{0,1\}^{D \cdot n_1 \cdot n_2}$.

1: **for** $(i = 0; i < n_1; i \leftarrow i + 1)$ **do**
2: **for** $(j = 0; j < n_2; j \leftarrow j + 1)$ **do**
3: $\mathbf{C}_{i,j} = \sum_{l=0}^{D-1} c_l \cdot 2^l$ where $c_l \in \{0,1\}$
4: **for** $(l = 0; l < D; l \leftarrow l + 1)$ **do**
5: $\mathbf{b}_{(i \cdot n_2 + j)D + l} \leftarrow c_{D-1-l}$
6: **return** $\mathbf{b}$

Algorithm 12 Frodo.Unpack

Input: Bit string $\mathbf{b} \in \{0,1\}^{D \cdot n_1 \cdot n_2}$, n_1, n_2.
Output: Matrix $\mathbf{C} \in \mathbb{Z}_q^{n_1 \times n_2}$.

1: **for** $(i = 0; i < n_1; i \leftarrow i + 1)$ **do**
2: **for** $(j = 0; j < n_2; j \leftarrow j + 1)$ **do**
3: $\mathbf{C}_{i,j} \leftarrow \sum_{l=0}^{D-1} \mathbf{b}_{(i \cdot n_2 + j)D + l} \cdot 2^{D-1-l}$
4: **return** $\mathbf{C}$

C Security reductions

FrodoKEM depends on the hardness of plain learning with errors. A summary of the reductions supporting the security of FrodoKEM is as follows:

1. FrodoKEM, using the concrete error distributions χ_{Frodo} specified in Table 3, is an IND-CCA-secure KEM against classical attacks in the classical random oracle model, under the assumption that FrodoPKE using a rounded Gaussian error distribution is an IND-CPA-secure public-key encryption scheme against classical attacks. This is Theorem 1, and the reduction is tight.
2. FrodoKEM, using any error distribution, is an IND-CCA-secure KEM against quantum attackers in the quantum random oracle model, under the assumption that FrodoPKE using the same error distribution is an IND-CPA-secure public-key encryption scheme against quantum attackers. This is Theorem 2, and the reduction is non-tight. We view this theorem as supporting the security of general constructions of LWE-based KEMs in the style of FrodoKEM against quantum adversaries, but it does not give meaningful *concrete* quantum bit-security estimates for the twelve FrodoKEM instantiations in this document, which is why we omit the corresponding column from Table 4.
3. The IND_{n_c,n_u}-CCA security of FrodoKEM reduces tightly to the IND_{n_c,n_u}-CPA security of FrodoPKE. The argument is found in [GHS25].
4. Changing the distribution of matrix $\mathbf{A}$ from a truly uniform distribution to one generated from a public random seed in a pseudorandom fashion does not affect the

security of FrodoKEM or FrodoPKE, provided that the pseudorandom generator is modeled either as an ideal cipher (when using AES128) or a random oracle (when using SHAKE128). This is shown in Section C.1.3.

5. FrodoPKE, using any error distribution and a uniformly random $\mathbf{A}$, is an IND_{n_c,n_u}-CPA-secure public-key encryption scheme under the assumption that the uniform-secret learning with errors decision problem is hard for the same parameters (except for a small additive loss in the number of samples), for either classical or quantum adversaries. This is a consequence of Theorem 3 and Theorem 4, and the result is tight.
6. The uniform-secret learning with errors decision problem, using a rounded Gaussian distribution with parameter σ from Table 3 and an appropriate bound on the number of samples, is hard under the assumption that the *worst-case* bounded-distance decoding with discrete Gaussian samples problem (BDDwDGS, Definition 25) is hard for related parameters. Theorem 5 gives a non-tight classical reduction against classical or quantum adversaries (in the standard model).

C.1 IND_{n_c,n_u}-CCA security reduction

Here we give a detailed description of the proof steps for Theorem 1.

Step 1: IND_{n_c,n_u}-CPA PKE to OW_{n_c,n_u}-PCA deterministic PKE_1. For completeness, we recall the definition of OW_{n_c,n_u}-PCA, following the presentation of Hofheinz et al. [HHK17, Höv21], and extended in [GHS25].

Definition 23 (OW_{n_c,n_u}-PCA for PKE [OP01])**.** Let PKE be a public-key encryption scheme and let $\mathcal{A}$ be an algorithm. The OW_{n_c,n_u}-PCA security experiment for $\mathcal{A}$ attacking PKE is $\mathrm{Exp}^{\mathsf{OW}_{n_c,n_u}\text{-PCA}}_{\mathsf{PKE}}(\mathcal{A})$ from Figure 6. The advantage of $\mathcal{A}$ in the experiment is

$$\mathrm{Adv}^{\mathsf{OW}_{n_c,n_u}\text{-PCA}}_{\mathsf{PKE}}(\mathcal{A}) := \Pr\left[\mathrm{Exp}^{\mathsf{OW}_{n_c,n_u}\text{-PCA}}_{\mathsf{PKE}}(\mathcal{A}) \Rightarrow 1\right].$$

Experiment $\mathrm{Exp}^{\mathsf{OW}_{n_c,n_u}\text{-PCA}}_{\mathsf{PKE}}(\mathcal{A})$:

for $j = 1, \ldots, n_u$ **do**
  $(pk_j, sk_j) \leftarrow \mathsf{PKE.KeyGen}()$
$\vec{pk} = (pk_1, ..., pk_{n_u})$
$(j, m') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{Pco}}(\cdot,\cdot),\mathsf{chall}(\cdot)}(\vec{pk})$
if $m' \in \mathfrak{L}_{M_j}$ **then**
  return 1
else
  return 0

Oracle $\mathcal{O}_{\mathrm{Pco}}(j, m, c)$:

if $\mathsf{PKE.Dec}(c, sk_j) = m$ **then**
  return 1
else
  return 0

Oracle $\mathsf{chall}(j)$:

[may be called up to n_c times]
$m \leftarrow_{\$} \mathcal{M}$
$\mathfrak{L}_{M_j} := \mathfrak{L}_{M_j} \cup \{m\}$
$c^* \leftarrow \mathsf{PKE.Enc}(m, pk_j)$
return c^*

Figure 6: Security experiment for OW_{n_c,n_u}-PCA.

The ST transform converts a public-key encryption scheme PKE to a deterministic public-key encryption scheme PKE_1; see Figure 7. Glabush's Theorem 4.2.3 tightly establishes the OW-PCVA-security of PKE_1 under, among others, the assumption that PKE is IND-CPA secure and γ-spread. (In the OW-PCVA security game, the attacker additionally has a ciphertext-validity oracle, which checks whether a queried ciphertext has a valid decryption.) These results were adapted to the multi-target setting in [GHS25]. However, they note that OW_{n_c,n_u}-PCA security follows (tightly) *without* the γ-spread assumption,

$\underline{\mathsf{PKE}_1.\mathrm{KeyGen}():}$
1: **return** $\mathsf{PKE}.\mathrm{KeyGen}()$

$\underline{\mathsf{PKE}_1.\mathrm{Enc}(\mu, pk):}$
1: $\mathsf{salt} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$
2: $r \leftarrow G_2(\mu \| \mathsf{salt})$
3: $c \leftarrow \mathsf{PKE}.\mathrm{Enc}(\mu, pk; r)$
4: **return** $c \| \mathsf{salt}$

$\underline{\mathsf{PKE}_1.\mathrm{Dec}(c \| \mathsf{salt}, sk):}$
1: $\mu' \leftarrow \mathsf{PKE}.\mathrm{Dec}(c, sk)$
2: **if** $\mu' = \perp$ or $c \neq \mathsf{PKE}.\mathrm{Enc}(\mu', pk; G_2(\mu' \| \mathsf{salt}))$ **then**
3: **return** $\perp$
4: **else**
5: **return** μ'

Figure 7: Construction of deterministic public-key encryption scheme $\mathsf{PKE}_1 = \mathsf{ST}[\mathsf{PKE}, G_2]$ from a public-key encryption scheme PKE and hash function G_2.

because in the security bounds γ-spreadness is relevant only to ciphertext-validity queries. We state that adapted version here.

Lemma 2 ([GHS25], Theorem 23, OW_{n_c,n_u}-PCA version)**.** *Let* PKE *be a* $\delta(n_u)$*-correct public-key encryption scheme with message space* $\mathcal{M}$*. For any* OW-PCA *adversary* $\mathcal{A}$ *that issues at most* q_G *queries to the random oracle* G_2 *and* q_P *queries to the plaintext-checking oracle, there exists an* IND-CPA *adversary* $\mathcal{B}$ *such that,*

$$\mathrm{Adv}^{\mathsf{OW}_{n_c,n_u}\text{-PCA}}_{\mathsf{PKE}_1}(\mathcal{A}) \leq q_G \cdot \delta(n_u) + \frac{6q_G}{|\mathcal{M}|} + \frac{1}{2^\lambda} + \frac{n_c}{|\mathcal{M}|} + 2 \cdot \mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-CPA}}_{\mathsf{PKE}}(\mathcal{B}) \ ,$$

where $\lambda \geq 512$ *for all* FrodoKEM *parameter sets; and the running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$ *plus the time needed to simulate the random oracle.*

It is straightforward to verify from the proof that $\mathcal{B}$ uses $\mathcal{A}$ solely as a "black box" subroutine.

Step 2: Approximating the error distribution. The rounded Gaussian distribution (Definition 16), which is important to the worst-case-to-average-case reduction, is difficult to sample on a finite computer (and impossible to sample in constant time). Following Langlois et al. [LSS14], we replace this infinite-precision distribution with a finite approximation, and quantify the OW_{n_c,n_u}-PCA security loss using their Rényi divergence.

Definition 24 (Rényi divergence)**.** The Rényi divergence of positive order $\alpha \neq 1$ of a discrete distribution P from a distribution Q is defined as

$$\mathrm{D}_\alpha(P\|Q) = \frac{1}{\alpha - 1} \ln\left(\sum_{x \in \mathrm{supp}\, P} P(x) \left(\frac{P(x)}{Q(x)} \right)^{\alpha - 1} \right) \ .$$

Note that our definition differs from that of [LSS14] in that we take the logarithm of the sum, and that Rényi divergence is not symmetric. The following result relates probabilities of a certain event occurring under two distributions as a function of their Rényi divergence.

Lemma 3 ([LSS14, Lemma 4.1])**.** *Let* S *be an event defined in a probabilistic experiment* G_Q *in which* s *samples are drawn from distribution* Q*. Then the probability that* S *occurs in the same experiment but with* Q *replaced by* P *is bounded as follows:*

$$\Pr[G_P(S)] \leq \left(\Pr[G_Q(S)] \cdot \exp(s \cdot \mathrm{D}_\alpha(P\|Q))\right)^{1-1/\alpha} . \tag{6}$$

It immediately follows that reductions from any *search* problem, such as the one represented by the OW_{n_c,n_u}-PCA game, are preserved up to the relaxation in (6). For any given security relationship, and any concrete choice of the two distributions P and Q, the loss can be minimized by choosing an optimal value of the order α.

Corollary 1 (Distribution substitution for OW_{n_c,n_u}-PCA)**.** *Let* PKE_X *be a public-key encryption scheme that is parameterized by a distribution* X*, and let* s *be an upper bound on the total number of samples drawn from* X *by* PKE_X.Enc *and* PKE_X.KeyGen *combined. Let* $\mathcal{A}$ *be an* OW-PCA *adversary against* PKE_X*, and let* P *and* Q *be discrete distributions. Then for any* $\alpha > 1$*,*

$$\mathrm{Adv}^{\mathsf{OW}_{n_c,n_u}\text{-PCA}}_{\mathsf{PKE}_P}(\mathcal{A}) \leq \left(\mathrm{Adv}^{\mathsf{OW}_{n_c,n_u}\text{-PCA}}_{\mathsf{PKE}_Q}(\mathcal{A}) \cdot \exp(s \cdot \mathrm{D}_\alpha(P\|Q))\right)^{1-1/\alpha} .$$

Proof. This follows immediately from Lemma 3, with S being the event that $\mathcal{A}$ "wins" the OW-PCA experiment from Figure 6, i.e., causes it to output 1. □

We use Corollary 1 to relate the OW-PCA security of $\mathsf{T}[\mathsf{FrodoPKE}_P, G_2]$ to the OW-PCA security of $\mathsf{T}[\mathsf{FrodoPKE}_\Psi, G_2]$ where $\mathsf{FrodoPKE}_\Psi$ is the same as FrodoPKE but with the error distribution $P = \chi_{\mathsf{Frodo}}$ replaced by a rounded Gaussian distribution $Q = \Psi$ (see Definition 16).

Step 3: OW_{n_c,n_u}-PCA deterministic PKE_1 to IND_{n_c,n_u}-CCA KEM. Hofheinz et al. [HHK17] define the $\mathsf{U}^{\not\perp}$ transform from a deterministic public-key encryption scheme PKE_1 to a key encapsulation mechanism $\mathsf{KEM}^{\not\perp}$; see Figure 8. Hofheinz et al.'s Theorem 3.4 shows the IND-CCA security of $\mathsf{KEM}^{\not\perp} = \mathsf{U}^{\not\perp}[\mathsf{PKE}_1, F]$ assuming the OW-PCA security of the underlying PKE_1. This result was adapted to the target setting in Theorem 24 from [GHS25]. This result is stated below in Lemma 4.

$\underline{\mathsf{KEM}^{\not\perp}.\mathrm{KeyGen}()}$:
1: $(pk, sk) \leftarrow\!\!\$\ \mathsf{PKE}_1.\mathrm{KeyGen}()$
2: $s \leftarrow\!\!\$\ \mathcal{M}$
3: $sk' \leftarrow (sk, s)$
4: **return** (pk, sk')

$\underline{\mathsf{KEM}^{\not\perp}.\mathrm{Encaps}(pk)}$:
1: $\mathsf{salt} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$
2: $c\|\mathsf{salt} \leftarrow \mathsf{PKE}_1.\mathrm{Enc}(\mu, pk)$
3: $ss \leftarrow F(\mu, c\|\mathsf{salt})$
4: **return** $(c\|\mathsf{salt}, ss)$

$\underline{\mathsf{KEM}^{\not\perp}.\mathrm{Decaps}(c\|\mathsf{salt}, (sk, s))}$:
1: $\mu' \leftarrow \mathsf{PKE}.\mathrm{Dec}(c\|\mathsf{salt}, sk)$
2: **if** $\mu' \neq \perp$ **then**
3: **return** $ss' \leftarrow F(\mu', c\|\mathsf{salt})$
4: **else**
5: **return** $ss' \leftarrow F(s, c\|\mathsf{salt})$

Figure 8: Construction of key encapsulation mechanism $\mathsf{KEM}^{\not\perp} = \mathsf{U}^{\not\perp}[\mathsf{PKE}_1, F]$ from a deterministic public-key encryption scheme PKE_1 and hash function F.

Lemma 4 ([GHS25, Theorem 24])**.** *Model* F *as a random oracle. Then if* PKE_1 *is* δ_1*-correct, so is* $\mathsf{KEM}^{\not\perp}$*. For any* IND-CCA *adversary* $\mathcal{A}$ *against* $\mathsf{KEM}^{\not\perp}$ *issuing at most* q_F *queries to* F*, there exists an* OW-PCA *adversary* $\mathcal{B}$ *against* PKE_1 *that makes at most* q_F *queries to its plaintext-checking oracle, and for which*

$$\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-CCA}}_{\mathsf{KEM}^{\not\perp}}(\mathcal{A}) \leq \frac{2n_u n_c(n_c-1)}{|\mathcal{M}|2^{\mathsf{len}_{\mathsf{salt}}}} + \frac{2q_F}{|\mathcal{M}|} + 2\delta(n_u) + 2\mathrm{Adv}^{\mathsf{OW}_{n_c,n_u}\text{-PCA}}_{\mathsf{PKE}_1}(\mathcal{B}) ,$$

where the running time of $\mathcal{B}$ *is about that of* $\mathcal{A}$*, plus the time to simulate the random oracle and decapsulation queries.*

It is straightforward to verify from the proof that $\mathcal{B}$ uses $\mathcal{A}$ solely as a "black box" subroutine. Together, Lemma 2, Corollary 1, and Lemma 4 establish Theorem 1.

C.1.1 Applying Theorem 1 in the single-public key, single-ciphertext setting

For an application of Theorem 1 to our schemes, consider the relation between the IND-CCA security of FrodoKEM-640 and the IND-CPA security of FrodoPKE-640$_\Psi$, where the error distribution of the latter is taken to be the rounded Gaussian $\Psi_{2.8\cdot\sqrt{2\pi}}$ as defined in Section A.2. To extract exact bounds on the IND-CCA security (in the classical ROM) of FrodoKEM-640, we use Theorem 1 setting $n_\mathsf{c} = n_\mathsf{u} = 1$, meaning IND-CCA = IND$_{1,1}$-CCA, and we make a number of assumptions about the underlying cost model. Specifically,

- We ignore the overhead of running the reduction of Theorem 1, including the cost of simulating random oracles.
- We bound the cost to the adversary of *making* an oracle query as $\geq 2^{18}$ classical gates. This bound is based on the NIST Call for Proposals, Section 4.A.5, which estimates the cost of finding collisions in SHA-3 at all security levels. (Here we ignore the small performance differences between SHAKE128, SHAKE256, and SHA3-256.)
- We interpret "b bits of classical security" as a statement that the advantage in the corresponding game of a uniform t-gate classical adversary is bounded by $t/2^b$. For some tasks, such as collision finding, this upper bound can be quite loose for smaller values of t (and thus beneficial to the adversary).
- We assume the IND-CPA bit-security of FrodoPKE-640$_\Psi$ to be given by the smaller of the costs of the primal and dual attacks on the LWE problem (Table 6), discounted by the reduction factor of $\overline{n} + \overline{m} = 16$ (Theorem 3), yielding $2^{134.5}$ in the single-user, single-ciphertext setting.

Under these assumptions, if an adversary $\mathcal{B}$ has uniform gate complexity t, then it has advantage $\mathrm{Adv}^{\text{IND-CPA}}_{\text{FrodoPKE-640}_\Psi}(\mathcal{B}) = \mathrm{Adv}^{\text{IND}_{1,1}\text{-CPA}}_{\text{PKE}_Q}(\mathcal{B})$ bounded by $t \cdot 2^{-134.5}$.

The Rényi divergence of $\chi_{\text{Frodo-640}}$ from the rounded Gaussian distribution is given by $\mathrm{D}_\alpha(\chi_{\text{Frodo-640}} \| \Psi_{2.8\cdot\sqrt{2\pi}}) \leq 0.0000324$ for $\alpha = 200$ (Table 3). The number of samples drawn from the error distribution by FrodoPKE.KeyGen is $2n\overline{n}$, and by FrodoPKE.Enc is $2\overline{m}n + \overline{m}\overline{n}$, which for $n = 640$ and $\overline{m} = \overline{n} = 8$ totals $s = 2 \times (8+8) \times 640 + 64 = 20544$.

We now recall Equation 3 from Theorem 1:

$$\mathrm{Adv}^{\text{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-CCA}}_{\text{KEM}^{\not\perp\prime}_P}(\mathcal{A}) \leq 2 \cdot \left(\left(\left(\frac{6q_\mathsf{RO}}{|\mathcal{M}|} + \frac{1}{2^{512}} + \frac{n_\mathsf{c}}{|\mathcal{M}|} + q_\mathsf{RO} \cdot \delta(n_\mathsf{u}) + 2 \cdot \mathrm{Adv}^{\text{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-CPA}}_{\text{PKE}_Q}(\mathcal{B}) \right) \cdot \exp(s \cdot \mathrm{D}_\alpha(P\|Q)) \right)^{1-1/\alpha} + \delta(n_\mathsf{u}) + \frac{q_\mathsf{RO}}{|\mathcal{M}|} + \frac{n_\mathsf{u} n_\mathsf{c}(n_\mathsf{c}-1)}{|\mathcal{M}| \cdot 2^{\mathsf{len}_\mathsf{salt}}} \right).$$

To obtain an IND$_{n_\mathsf{c},n_\mathsf{u}}$-CCA bit-security value, we would upper-bound $\mathrm{Adv}^{\text{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-CCA}}_{\text{KEM}^{\not\perp\prime}_P}(\mathcal{A})/t$. We proceed to do this by observing that due to the gate complexity of implementing random oracles, for any adversary running in time $\leq t$, we have $q_\mathsf{RO} < t \cdot 2^{-18}$, and by assuming that any realistic IND$_{n_\mathsf{c},n_\mathsf{u}}$-CCA adversary $\mathcal{A}$ must run in time $t \geq 2^{32}$. Making these two observations, we obtain

$$\frac{\mathrm{Adv}^{\text{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-CCA}}_{\text{KEM}^{\not\perp\prime}_P}(\mathcal{A})}{t} \leq 2 \left(\left(\left(\begin{array}{l} \frac{6\cdot 2^{-18}}{|\mathcal{M}|} + 2^{-32}\frac{1}{2^{512}} \\ + 2^{-32}\frac{n_\mathsf{c}}{|\mathcal{M}|} + 2^{-18}\delta(n_\mathsf{u}) \\ + 2 \cdot \frac{\mathrm{Adv}^{\text{IND}_{n_\mathsf{c},n_\mathsf{u}}\text{-CPA}}_{\text{PKE}_Q}(\mathcal{B})}{t} \end{array} \right) \exp(s \cdot \mathrm{D}_\alpha(P\|Q)) \right)^{1-1/\alpha} + 2^{-32} \cdot \delta(n_\mathsf{u}) + \frac{2^{-18}}{|\mathcal{M}|} + 2^{-32} \cdot \frac{n_\mathsf{u} n_\mathsf{c}(n_\mathsf{c}-1)}{|\mathcal{M}| \cdot 2^{\mathsf{len}_\mathsf{salt}}} \right).$$

For a single user and single challenge ciphertext, we set $n_\mathsf{c} = n_\mathsf{u} = 1$, and the above

simplifies to

$$\frac{\mathrm{Adv}^{\mathsf{IND}_{1,1}\text{-}\mathsf{CCA}}_{\mathsf{KEM}^{\not\perp}_{P}}(\mathcal{A})}{t} \leq 2\Bigg(\Bigg(\Bigg(\frac{2^{-15}+2^{-32}}{|\mathcal{M}|} + 2^{-32}\frac{1}{2^{512}} + 2^{-18}\cdot\delta(1) + 2\frac{\mathrm{Adv}^{\mathsf{IND}_{1,1}\text{-}\mathsf{CPA}}_{\mathsf{PKE}_Q}(\mathcal{B})}{t}\Bigg) \cdot \exp(s\cdot \mathrm{D}_\alpha(P\|Q))\Bigg)^{1-1/\alpha} + 2^{-32}\delta(1) + \frac{2^{-18}}{|\mathcal{M}|}\Bigg). \tag{7}$$

We can bound the advantage in $\mathrm{Exp}^{\mathsf{IND}_{1,1}\text{-}\mathsf{CCA}}_{\mathsf{FrodoKEM\text{-}640}}$ for an adversary $\mathcal{A}$ against FrodoKEM-640 with gate count $t \geq 2^{32}$ by substituting $|\mathcal{M}| = 2^{128}$ and $\delta(1) < 2^{-138.7}$ into Equation 7, as follows:[13]

$$\begin{aligned}\frac{\mathrm{Adv}^{\mathsf{IND}_{1,1}\text{-}\mathsf{CCA}}_{\mathsf{FrodoKEM\text{-}640}}(\mathcal{A})}{t} &\leq 2\cdot\Bigg(\Bigg(\Bigg(\frac{2^{-15}+2^{-32}}{2^{128}} + 2^{-512-32} + 2^{-18}\cdot 2^{-138.7} + 2\cdot 2^{-134.5}\Bigg) \\ &\qquad \cdot \exp(20544\cdot 0.0000324)\Bigg)^{0.995} + 2^{-32}\cdot 2^{-138.7} + \frac{2^{-18}}{2^{128}}\Bigg) \\ &\leq 2^{-130.876}.\end{aligned}$$

Similarly, computed bounds on the advantage of a classical single-user, single-ciphertext IND-CCA adversary for other parameter settings appear in Table 4.

C.1.2 Multi-user multi-ciphertext concrete security

Concrete bit-security estimates can be obtained in a similar manner to Section C.1.1, using larger values for n_{c} and n_{u}. It is less clear what the "target" bit-security values would be for each NIST security category. In the single-user, single-ciphertext setting, NIST's suggestion [Nat17] was to compare bit-security with the gate-cost of key-search on AES given message-ciphertext pairs. One natural possibility would be to compare FrodoKEM's bit security against the multi-key, multi-evaluation security of a block cipher (such as AES) as a PRF.

Single-key, multi-ciphertext security. Applying the PRP-PRF switching lemma to AES, the multi-ciphertext advantage upper-bound degrades by an additive term $q(q-1)/2^{128}$, where q is the number of AES evaluations. In the case of FrodoKEM, the $\mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},1}\text{-}\mathsf{CCA}}_{\mathsf{FrodoKEM\text{-}640}}$ advantage upper-bound in Theorem 1 degrades from $\mathrm{Adv}^{\mathsf{IND}_{n_{\mathsf{c}},1}\text{-}\mathsf{CPA}}_{\mathsf{PKE}_Q}$ by an additive term bounded above as a function of n_{c} by

$$\mathcal{O}\left(\frac{n_{\mathsf{c}}}{|\mathcal{M}|}\cdot\exp(s\cdot\mathrm{D}_\alpha(P\|Q)) + \frac{n_{\mathsf{c}}(n_{\mathsf{c}}-1)}{|\mathcal{M}|\cdot 2^{\mathsf{len}_{\mathsf{salt}}}}\right) = \mathcal{O}\left(\frac{n_{\mathsf{c}}^2}{|\mathcal{M}|\cdot 2^{\mathsf{len}_{\mathsf{salt}}}}\right).$$

The advantage degrades further at the IND-CPA level, as the adversary is given n_{c} different LWE challenges, one per encapsulation. While this implies a linear degradation in the advantage upper bound, $\mathrm{Adv}^{\mathsf{IND}_{1,n_{\mathsf{c}}}\text{-}\mathsf{CPA}}_{\mathsf{PKE}_Q} \leq n_{\mathsf{c}}\cdot\mathrm{Adv}^{\mathsf{IND}_{1,1}\text{-}\mathsf{CPA}}_{\mathsf{PKE}_Q}$, we are not aware of any attack on LWE, lattice reduction or otherwise, that gains such an advantage by being able to pick among n_{c} different instances.

Overall for eFrodoKEM, this degradation is similar to that of AES as a PRF, with the multi-ciphertext advantage upper bound growing as $O(n_{\mathsf{c}}^2/|\mathcal{M}|)$ in n_{c}. Consequently, a salt has been added to the definition of FrodoKEM, in order to achieve better-than-birthday-bound security in the single-key, multi-ciphertext setting (c.f. Table 2).

[13] This bound on $\delta(1)$ was obtained during parameter search for Frodo, and can be computed with our code release.

Multi-key security. Generic multi-key security bounds for PRFs see the advantage degrade by a linear factor k, where k is the number of keys under which the adversary can request PRF evaluations [BCK96]. In the case of FrodoKEM, due to hashing the public key at the moment of generating $\mathsf{seed}_{\mathbf{SE}} \| \mathbf{k}$ within encapsulation, the advantage in Theorem 1 degrades from $\mathrm{Adv}^{\mathsf{IND}_{n_c,n_u}\text{-}\mathsf{CPA}}_{\mathsf{PKE}_Q}$ by an additive term bounded above as a function of n_u by

$$\mathcal{O}\left((q_{\mathsf{RO}} \cdot \exp(s \cdot \mathrm{D}_\alpha(P\|Q)) + 1) \cdot \delta(n_u) + \frac{n_c(n_c-1)}{|\mathcal{M}| \cdot 2^{\mathsf{len}_{\mathsf{salt}}}} \cdot n_u\right).$$

The gap is linear in n_u as $\delta(n_u) \leq \delta(1) \cdot n_u$ [DHK+21], and is practically tight when using a long enough salt and whenever $\delta(n_u) \ll \delta(1) \cdot n_u$; as mentioned in Section 2.2, there is evidence for the latter for KEMs built from lattice-based PKE (see [DHK+21, Table 1]).

At the IND-CPA level, similarly to the single-key, multi-ciphertext setting, degradation is linear in n_u since n_u different LWE challenges are provided to the adversary in the form of public keys. Again, we are not aware of any practical improvement of attacks on LWE achievable by considering multiple independent instances.

C.1.3 Deterministic generation of A

The matrix $\mathbf{A}$ in FrodoKEM and FrodoPKE is deterministically expanded from a short random seed in the function Frodo.Gen either using AES128 or SHAKE128. In order to relate FrodoKEM and FrodoPKE's security to the hardness of the learning with errors problem, it was argued in the Round 3 submission of FrodoKEM [NAB+20] that one can replace a uniformly sampled $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$ with matrices sampled according to Frodo.Gen. Although the matrix appears pseudorandom under standard security assumptions to an adversary without access to the seed, [NAB+20] argues security of this step against a stronger (and more realistic) adversary via the indifferentiability framework [MRH04, CDMP05]. Refer to [NAB+20] for complete details.

C.1.4 IND-CPA security

In this section we show that FrodoPKE, using any error distribution χ and uniformly random $\mathbf{A}$, is an IND-CPA-secure public-key encryption scheme based on the hardness of the learning with errors decision problem with the same error distribution. We first tightly relate the IND-CPA security of FrodoPKE to the normal-form DLWE problem, where the secret coordinates have the same distribution as the errors.

Theorem 3 (normal-form DLWE $\implies$ IND-CPA security of FrodoPKE)**.** *Let $n, q, \overline{m}, \overline{n}$ be positive integers, and χ be a probability distribution on $\mathbb{Z}$. There exist classical algorithms $\mathcal{B}_1, \mathcal{B}_2$ that use as a "black box" subroutine any (quantum or classical) algorithm $\mathcal{A}$ against the* IND-CPA *security of* FrodoPKE *(with a uniformly random $\mathbf{A}$), for which*

$$\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{FrodoPKE}}(\mathcal{A}) \leq \overline{n} \cdot \mathrm{Adv}^{\mathsf{nf\text{-}dlwe}}_{n,n,q,\chi}(\mathcal{B}_1) + \overline{m} \cdot \mathrm{Adv}^{\mathsf{nf\text{-}dlwe}}_{n,n+\overline{n},q,\chi}(\mathcal{B}_2) \ .$$

The running times of $\mathcal{B}_1$ and $\mathcal{B}_2$ are approximately that of $\mathcal{A}$.

The proof of Theorem 3 is the same as that of [LP11, Theorem 3.2] or [BCD+16, Theorem 5.1].

The following theorem relates the LWE decision problem in its normal form to one where the secret is *uniformly random* over $\mathbb{Z}_q$. We need this only for connecting the latter variant, which arises in the reduction from worst-case lattice problems described in the next subsection, to the normal form as used in FrodoPKE. (In particular, our cryptanalysis and concrete security bounds are for the normal form.) The theorem is specialized to power-of-two modulus q (our case of interest), and the stated bounds in the advantage and number of LWE samples are more precise than those given in the original work. These bounds follow from the fact that, by a straightforward calculation, a uniformly random

n-by-$(n+k)$ matrix over $\mathbb{Z}_q$ has an invertible n-by-n submatrix except with probability at most 2^{-k}.

Theorem 4 (uniform-secret DLWE $\implies$ normal-form DLWE; [ACPS09], Lemma 2)**.** *Let n, m, k, q be positive integers with $q \geq 2$ a power of two, and let χ be a probability distribution on $\mathbb{Z}$. There exists a classical algorithm $\mathcal{B}$ that uses as a "black box" subroutine any (quantum or classical) algorithm $\mathcal{A}$ against the normal-form LWE decision problem, for which*

$$\mathrm{Adv}^{\mathsf{nf\text{-}dlwe}}_{n,m,q,\chi}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{dlwe}}_{n,m+n+k,q,\chi}(\mathcal{B}) + 2^{-k} \quad .$$

The running time of $\mathcal{B}$ is approximately that of $\mathcal{A}$.

C.1.5 Reductions from worst-case lattice problems

When choosing parameters for LWE, one needs to choose an error distribution, and in particular its "width." Certain choices (e.g., sufficiently wide Gaussians) are supported by *reductions* from worst-case lattice problems to LWE; see, e.g., [Reg09, Pei09a, BLP+13, PRS17]. At a high level, such a reduction transforms any algorithm that solves LWE *on the average*—i.e., for random instances sampled according to the prescribed distribution—into an algorithm of related efficiency that solves *any instance* of certain lattice problems (not just random instances).

The original work of [Reg09] and a follow-up work [PRS17] gave quantum polynomial-time reductions, from the worst-case GapSVP_γ (Definition 20), SIVP_γ (Definition 21), and DGS_φ (Definition 22) problems on n-dimensional lattices, to n-dimensional LWE (for an unbounded polynomial $m = \mathrm{poly}(n)$ number of samples) with Gaussian error of standard deviation $\sigma \geq c\sqrt{n}$. The constant factor c was originally stated as $c = \sqrt{2/\pi}$, but can easily be improved to any $c > 1/(2\pi)$ via a tighter analysis of essentially the same proof.[14] However, for efficiency reasons our choices of σ (see Table 3) are somewhat smaller than what is required by these reductions.

Instead, following [Reg09, Section 1.1], below we obtain an alternative *classical* (i.e., non-quantum) reduction from a variant of the worst-case bounded-distance decoding (BDD) problem to our LWE parameterizations. In contrast to the quantum reductions described above, which requires Gaussian error of standard deviation $\sigma \geq c\sqrt{n}$, the alternative reduction supports a smaller error width—as small as the "smoothing parameter" [MR07] of the lattice of integers $\mathbb{Z}$. For the BDD variant we consider, which we call "BDD with Discrete Gaussian Samples" ($\mathsf{BDDwDGS}$), the input additionally includes discrete Gaussian samples over the dual lattice, but having a larger width than known algorithms are able to exploit [LLM06, DRS14]. This gives the problem a "hybrid" worst-case/average-case nature: it is worst case over the choice of the lattice, but average-case over the choice of the DGS samples. Details follow.

Bounded-distance decoding with discrete Gaussian samples. We first define a variant of the bounded-distance decoding problem, which is implicit in prior works that consider "BDD with preprocessing," [AR05, LLM06, DRS14] and recall the relevant aspects of known algorithms for the problem.

Definition 25 (Bounded-distance decoding with discrete Gaussian samples)**.** For a lattice $\mathcal{L} \subset \mathbb{R}^n$ and positive reals $d < \lambda_1(\mathcal{L})/2$ and $r > 0$, an instance of the *bounded-distance decoding with discrete Gaussian samples* problem $\mathsf{BDDwDGS}_{\mathcal{L},d,r}$ is a point $\mathbf{t} \in \mathbb{R}^n$ such that $\mathrm{dist}(\mathbf{t}, \mathcal{L}) \leq d$, and access to an oracle that samples from $D_{\mathcal{L}^*,s}$ for any (adaptively) queried $s \geq r$. The goal is to output the (unique) lattice point $\mathbf{v} \in \mathcal{L}$ closest to $\mathbf{t}$.

[14] The approximation factor γ for GapSVP and SIVP is $\tilde{O}(qn/\sigma) = (qn/\sigma)\log^{O(1)} n$, and the parameter φ for DGS is $\Theta(q\sqrt{n}/\sigma)$ times the "smoothing parameter" of the lattice.

Remark 3. For a given distance bound d, known BDDwDGS algorithms use discrete Gaussian samples that all have the same width parameter s. However, the reduction to LWE will use the ability to vary s. Alternatively, we mention that when $r \geq \eta_\varepsilon(\mathcal{L}^*)$ for some very small $\varepsilon > 0$ (which will always be the case in our setting), we can replace the variable-width DGS oracle from Definition 25 with a fixed-width one that samples from $D_{\mathbf{w}+\mathcal{L}^*,r}$ for any queried coset $\mathbf{w} + \mathcal{L}^*$, always for the same width r. This is because we can use the latter oracle to implement the former one (up to statistical distance 8ε), by sampling $\mathbf{e}$ from the continuous Gaussian of parameter $\sqrt{s^2 - r^2}$ and then adding a sample from $D_{\mathcal{L}^*-\mathbf{e},r}$. See [Pei10, Theorem 3.1] for further details.

The state-of-the-art algorithms for solving BDDwDGS [AR05, LLM06, DRS14] employ a certain $\mathcal{L}$-periodic function $f_{\mathcal{L},1/r} \colon \mathbb{R}^n \to [0,1]$, defined as

$$f_{\mathcal{L},1/r}(\mathbf{x}) := \frac{\rho_{1/r}(\mathbf{x}+\mathcal{L})}{\rho_{1/r}(\mathcal{L})} = \mathop{\mathbb{E}}_{\mathbf{w}\sim D_{\mathcal{L}^*,r}} [\cos(2\pi\langle \mathbf{w}, \mathbf{x}\rangle)] \ , \tag{8}$$

where the equality on the right follows from the Fourier series of $f_{\mathcal{L},1/r}$ (see [AR05]). To solve BDDwDGS for a target point $\mathbf{t}$, the algorithms use several discrete Gaussian samples $\mathbf{w}_i \sim D_{\mathcal{L}^*,r}$ to estimate the value of $f_{\mathcal{L},1/r}$ at $\mathbf{t}$ and nearby points via Equation 8, to "hill climb" from $\mathbf{t}$ to the nearest lattice point. For the relevant points $\mathbf{t}$ we have the (very sharp) approximation $f_{\mathcal{L},1/r}(\mathbf{t}) \approx \exp(-\pi r^2 \cdot \mathrm{dist}(\mathbf{t},\mathcal{L})^2)$, so by the Chernoff-Hoeffding bound, approximating $f_{\mathcal{L},1/r}(\mathbf{t})$ to within (say) a factor of two uses at least

$$\frac{1}{f_{\mathcal{L},1/r}(\mathbf{t})^2} \approx \exp(2\pi r^2 \cdot \mathrm{dist}(\mathbf{t},\mathcal{L})^2)$$

samples.[15] Note that without enough samples, the "signal" of $f_{\mathcal{L},1/r}(\mathbf{t})$ is overwhelmed by measurement "noise," which prevents the hill-climbing from making progress toward the answer.

In summary, when limited to N discrete Gaussian samples, the known approaches to solving BDDwDGS are limited to distance

$$\mathrm{dist}(\mathbf{t},\mathcal{L}) \leq r^{-1}\sqrt{\ln(N)/(2\pi)} \ . \tag{9}$$

Having such samples does not appear to provide any speedup in decoding at distances that are larger than this bound by some constant factor greater than one. In particular, if $d \cdot r \geq \omega(\sqrt{\log n})$ (which is the smoothing parameter of the integer lattice $\mathbb{Z}$ for negligible error ε), then having $N = \mathrm{poly}(n)$ samples does not seem to provide any help in solving $\mathsf{BDDwDGS}_{\mathcal{L},d,r}$ (versus having no samples at all).

Reduction from BDDwDGS to LWE. We now recall the following result from [PRS17], which generalizes a key theorem from [Reg09] to give a reduction from BDDwDGS to the LWE decision problem.

Theorem 5 (BDDwDGS hard $\implies$ decision-LWE hard [PRS17, Lemma 5.4])**.** *Let $\varepsilon = \varepsilon(n)$ be a negligible function and let $m = poly(n)$ and $C = C(n) > 1$ be arbitrary. There is a probabilistic polynomial-time (classical) algorithm that, given access to an oracle that solves $\mathsf{DLWE}_{n,m,q,\alpha}$ with non-negligible advantage and input a number $\alpha \in (0,1)$, an integer $q \geq 2$, a lattice $\mathcal{L} \subset \mathbb{R}^n$, and a parameter $r \geq Cq \cdot \eta_\varepsilon(\mathcal{L}^*)$, solves $\mathsf{BDDwDGS}_{\mathcal{L},d,r}$ using $N = m \cdot poly(n)$ samples, where $d = \sqrt{1 - 1/C^2} \cdot \alpha q/r$.*

Remark 4. The above statement generalizes the fixed choice of $C = \sqrt{2}$ in the original statement (inherited from [Reg09, Section 3.2.1]), using [Reg09, Corollary 3.10]. In particular, for any constant $\delta > 0$ there is a constant $C > 1$ such that $d = (1-\delta) \cdot \alpha q/r$.

[15] In fact, the algorithms need approximation factors much better than two, so the required number of samples is even larger by a sizable constant factor. However, the above crude bound will be sufficient for our purposes.

In particular, by Equation 9, if the Gaussian parameter αq of the LWE error sufficiently exceeds $\sqrt{\ln(N)/(2\pi)}$ (e.g., by a constant factor greater than one), then the $\mathsf{BDDwDGS}_{\mathcal{L},d,r}$ problem is plausibly hard (in the worst case), hence so is the corresponding LWE problem from Theorem 5 (on the average). An interesting direction is to obtain a more precise bound on, and improve, the "sample overhead" of the reduction, i.e., the $\mathrm{poly}(n)$ factor connecting the number of LWE samples m and the number of DGS samples N.