A Practical Randomized Nearest-Colattice Framework for Arbitrary Norms

Wenwen Xia[1,3], Wessel van Woerden[2], Geng Wang[3⋆], and Dawu Gu[3⋆]

[1] Univ Bordeaux, CNRS, Inria, Bordeaux INP, IMB, UMR 5251, Talence, France, wenwen.xia@math.u-bordeaux.fr
[2] PQShield, the Netherlands, wessel.vanwoerden@pqshield.com
[3] Shanghai Jiao Tong University, China, {xww_summer,wanggxx, dwgu}@sjtu.edu.cn

Abstract. The approximate Closest Vector Problem (CVP) with large approximation factors, such as a polynomial approximation, arises in the concrete cryptanalysis of lattice-based signatures, yet it is commonly estimated indirectly through SVP after Kannan's embedding (Wang et al, PQC2025). Such reductions may return short vectors that do not encode valid CVP solutions and are less suited to reusable preprocessing, batched targets, and list output.

We give, to our knowledge, the first practical instantiation of the Nearest-Colattice framework (Espitau et al, ANTS2020). Our RandNColattice algorithm realizes quotient-lattice CVP calls on explicitly projected basis blocks using randomized slicer oracles (Ducas et al, PKC2020), and therefore naturally propagates multiple candidate branches for batches of targets. We further extend the randomized slicer with dimensions for free for both BDD and approximate CVP/ListCVP, and introduce HybridRandNColattice, which replaces the oracle on the final block with Klein's sampler to generate diverse randomized residual targets efficiently. The final candidate-filtering interface is norm-generic, while our concrete analysis focuses on the Euclidean and infinity norms. We derive blockwise distance and runtime models and construct an estimator over 4 strategy models: D4f-CVP, Prob-BDD, CVP_{γ_2}, and $\mathsf{Flexd4f\text{-}CVP}_{\gamma_2}$ with $1 \le \gamma_2 \le \sqrt{4/3}$. The estimator instantiates costs under a theoretical model and the AGPS20 and MATZOV22 gate-count models, and we apply it to non-amortized forgery estimates for Dilithium, HuFu, and one-more-ISIS. Finally, we implement HybridRandNColattice by extending the G6K-CPU based randomized slicer (Alexander et al, AsiaCrypt 2025). Our experiments support the Euclidean distance model, while showing that the spherical approximation used for infinity-norm filtering is optimistic because the actual blockwise error distribution is anisotropic. These results provide a concrete foundation for direct approximate-CVP cryptanalysis and identify tighter infinity-norm modeling as an important open problem.

Keywords: cryptanalysis, approximate CVP, Nearest-Colattice, sieve, slicer

⋆ Corresponding author.

1 Introduction

Formally, given a lattice $\mathcal{L} \subset \mathbb{R}^d$ and a target vector $\mathbf{t} \in \mathbb{R}^d$, the approximate CVP with approximation factor γ (denoted CVP_γ) asks for a lattice vector $\mathbf{v} \in \mathcal{L}$ such that:

$$\|\mathbf{v} - \mathbf{t}\| \leq \gamma \cdot \lambda_1(\mathcal{L}),$$

where $\lambda_1(\mathcal{L})$ denotes the Euclidean norm of the shortest vector in the lattice $\mathcal{L}$. Despite its foundational importance, the study of approximate CVP remains relatively limited compared to other lattice problems such as the Shortest Vector Problem (SVP) or the Learning with Errors (LWE) problem. Most existing research focuses on CVP_γ with $\gamma = O(1)$ [SFS09, DLdW19, Laa20, DLvW20], leaving the complexity and practical implications of larger approximation factors comparatively underexplored. Addressing this gap is essential for both advancing theoretical understanding and providing more accurate security assessments of practical lattice-based schemes.

CVP_γ is often treated as SVP_γ via Kannan's embedding. While formal theoretical reductions have a large loss in the approximation radius, the lattice-cryptanalytic 'folklore' often simply treats them as having a similar difficulty. While this view has indeed often been confirmed in the unique decoding regime (such as for LWE), the story in the approximate regime is more nuanced. Simply combining Kannan's embedding with the state-of-the-art heuristic approximate (List)SVP algorithms, could either fail to give CVP solutions in practice, or could be suboptimal for many variants of CVP. For example, this approach could lead to the following concrete problems in practice:

- A found (approximate) short vector in Kannan's embedding lattice does not have to correspond to a CVP solution, making these algorithms potentially fail in practice.
- Only a single target can be embedded at a time, which makes the cost for Batch-CVP multiplicative.
- Efficient ListCVP typically only gives solutions from the same small sublattice coset, possibly leading to strong dependency issues when filtering on ℓ_∞-norm solutions.

As the state-of-the-art practical example, the restricted SVP paper [WXG25] employs Kannan's embedding technique to probabilistically reduce approximate CVP to approximate SVP. This reduction enabled the use of well-studied SVP-solving techniques to address approximate CVP instances, and therefore leads to competitive concrete cost estimates for CVP. Furthermore, Theorem 6.2 in [WXG25] derived probability formulas for the event that a short vector in the Euclidean norm is also short in the infinity norm, providing a theoretical basis for analyzing the success of such reductions. This methodology was applied to assess the security of the Dilithium signature scheme, bridging the gap between theoretical results and practical cryptographic evaluations. However, these theoretical foundations only apply to a heuristic modeling of the found vectors;

one that does not match to the real approximate SVP methods being used for the estimates. Indeed, concretely [WXG25, Theorem 6.1] claims that a fraction $6 - 4\sqrt{2} \approx 0.343$ of the short vectors corresponds to a CVP solutions. However, as an extreme counterexample in practice, the authors of [WXG25] pointed out that Restricted SVP fails for their blockwise solvers when the lattice dimension is larger than 70 because the initial reduction shifts all valid CVP solutions entirely outside the active search block $\mathbf{B}_{[0:\kappa)}$, making the 0.343 probability inapplicable. To avoid that, we consider the Nearest-Colattice framework which circumvents Kannan's embedding to fundamentally resolve this structural flaw and achieve a practical and scalable framework.

Prior and related work. The Nearest-Colattice algorithm (Alg. 2 in [EK20]) proposed the first theoretical framework for solving CVP_γ with polynomial γ. In detail, it solves the CVP_γ problem by performing a full BKZ-β_0 preprocessing on the lattice basis $\mathbf{B}$, then calls the Nearest-Colattice to find the approximate closest vector to $\mathbf{t}$, where β is a given parameter and $k = \lfloor d/\beta \rfloor$. [EK20] provided a theoretical foundation for tackling larger approximation factors, but its practical applicability remained uncertain due to the lack of experimental validation and concrete applications in real-world security assessments. The Nearest-Colattice approach demonstrated the feasibility of solving approximate CVP beyond the traditionally studied small factors, yet its implementation and performance in practical scenarios were left as open questions.

Contribution. In this work, we give a practical randomized instantiation of the Nearest-Colattice framework in Sec. 4. We instantiate its quotient-CVP calls with randomized slicer oracles operating on explicitly projected basis blocks; since these oracles use randomized coset shifts and may return multiple candidates, we call the resulting algorithm Randomized Nearest-Colattice (abbreviated as `RandNColattice`). To increase the randomness and decrease the time cost, we propose the Hybrid `RandNColattice` (abbreviated as `HybridRandNColattice`), which replaces the oracle on the tail block with Klein's sampler. We design 4 parameterization strategies, analyze the output distance and cost for the resulting framework in Sec. 5, build a concrete estimator for the Euclidean and infinity norms, apply it to Dilithium, HuFu, and one-more-ISIS in Sec. 6, and provide an implementation based on G6K in Sec. 7. Our main contributions are summarized as follows.

- **Dimensions for free for ListCVP.** We introduce `RandSlicerD4f`, a randomized slicer oracle equipped with the "dimensions for free" technique. We analyze its application in solving both BDD and $\mathsf{ListCVP}_\gamma$ ($1 \le \gamma \le \sqrt{4/3}$). The resulting framework seamlessly integrates projected ListCVP, Babai lifting, and final full-dimensional filtering, enabling the generation of multiple candidate solutions per target.
- **Propose the (Hybrid) Randomized Nearest-Colattice algorithm.** We propose the Randomized Nearest-Colattice algorithm (abbreviated as `RandNColattice`), a practical instantiation of the abstract recursion using `RandSlicerD4f`, which supports batched targets and multi-branch candidate

propagation. To further enhance candidate diversity and save the time cost, we propose HybridRandNColattice, which integrates Klein's sampler into the final-block. This hybrid approach efficiently expands the search space, enabling optimized full-lattice reconstruction and norm-specific filtering.

- **Norm-aware distance and strategy models.** We derive a blockwise Euclidean distance model and analyze the negligible contribution of the randomized tail block. Based on the available block oracles, we define five structured HybridRandNColattice strategy models: D4f-CVP, and Prob-BDD for the Euclidean norm, and CVP_{γ_2} and $\mathsf{Flexd4f\text{-}CVP}_{\gamma_2}$ for the infinity norm. For the latter, we combine the Euclidean output radius with a ball–cube intersection model and explicitly identify the directional bias omitted by this approximation.
- **A concrete estimator and cryptanalytic applications.** We construct an estimator designed to optimize the strategy parameters under various cost models. Specifically, it enumerates BKZ and HybridRandNColattice parameters to derive the optimal internal block strategy for the selected model. The total costs are decomposed into progressive BKZ preprocessing, one-time short-vector-list generation, and target-dependent HybridRandNColattice calls. We instantiate these costs using a theoretical model and the concrete AGPS20 and MATZOV22 gate-count models. We apply the estimator to the forgery problems underlying Dilithium, HuFu, and one-more-ISIS-based blind signatures.
- **Implementation and experimental validation.** We implement the algorithm HybridRandNColattice[4] in a C++/Cython/Python code base, extending the most efficient sieving-based `RandSlicer` from G6K [KKNM25].[5] Our experiments support the predicted Euclidean approximation factors. They also demonstrate that the current isotropic model for the infinity norm is optimistic, thereby isolating the main geometric limitation that must be addressed to obtain tighter infinity-norm estimates.

2 Preliminaries

2.1 Notions

We denote vectors by lower-case bold letters, e.g. $\mathbf{a}, \mathbf{b}, \mathbf{c}, ...$, and matrices by upper-case bold letters, e.g. $\mathbf{A}, \mathbf{B}, \mathbf{C}, ...$ For a matrix $\mathbf{B} = (\mathbf{b}_0, \cdots, \mathbf{b}_{d-1})$, we write $\mathbf{b}_i$ as its i-th column vector. The Euclidean norm of a vector $\mathbf{v} \in \mathbb{R}^m$ is denoted by $\|\mathbf{v}\|$. We write $\langle \cdot, \cdot \rangle$ for inner products and $\cdot$ for matrix-vector products. By abuse of notation we consider vectors to be row resp. column vectors depending on context, such that $\mathbf{v} \cdot \mathbf{A}$ and $\mathbf{A} \cdot \mathbf{v}$ are both meaningful. We write $\mathbf{0}_{m \times d}$ as the $m \times d$ all zero matrix. If the dimensions are clear from

[4] Our source code is publicly available at https://anonymous.4open.science/r/hybrid_randomized_nearest_colattice-27F4.

[5] Baseline repository: https://github.com/ElenaKirshanova/g6k_hybrid/tree/ac_artifact

context, we may omit the subscripts. $|\mathbf{B}|$ is denoted as the absolute value of the determinant of matrix $\mathbf{B}$. We define $[a:b) := \{a, a+1, ..., b-1\}$, $[d) = [0:d)$, $\mathcal{B}^d(\mathbf{v}, r) = \left\{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{v}\|_2 \leq r\right\}$ and denote $\mathcal{B}^d(r)$ as the ball with center $\mathbf{0}$. Let $[-B, B]^d = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x}\|_\infty \leq B\}$ denote the d-dimensional hypercube centered at the origin with side length $2B$, where $\|\mathbf{x}\|_\infty = \max_{0 \leq i \leq d-1} |x_i|$ is the infinity norm.

2.2 Probability and Geometry

In this subsection, we introduce the fundamental probabilistic and geometric definitions utilized throughout our analysis.

Definition 2.1. *Let $\mathcal{U}[a, b]$ be the uniform distribution on $[a, b]$.*

Definition 2.2. *Let $\mathcal{N}(\mu, \sigma)$ denote the Gaussian distribution with mean μ, standard deviation σ, and the Probability Density Function (PDF) $\rho_\mu(x) := \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$. We denote the Cumulative Distribution Function (CDF) of the standard Gaussian $\mathcal{N}(0, 1)$ by $\Phi(x) := \int_{-\infty}^{x} \rho_0(t)dt$. For higher dimensions $d \geq 2$, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ represents a d-dimensional multivariate Gaussian distribution with mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and a symmetric positive-definite covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$. Its PDF is defined as*

$$\rho_{\boldsymbol{\mu}, \boldsymbol{\Sigma}}(\mathbf{x}) := \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right), \tag{1}$$

where $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$.

Definition 2.3 (Beta function). *The Beta function, denoted as $\mathrm{Beta}(\alpha, \beta)$, is an integral function defined for strictly positive real parameters α and β. It is formally defined as:*

$$\mathrm{Beta}(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}.$$

A continuous random variable X follows a Beta distribution, denoted as $X \sim \mathrm{Beta}(\alpha, \beta)$, if it is supported on the interval $[0, 1]$ with the probability density function (PDF):

$$f(x; \alpha, \beta) = \frac{1}{\mathrm{Beta}(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}.$$

The cumulative distribution function (CDF) is the regularized incomplete Beta function, denoted as $I_x(\alpha, \beta)$.

2.3 Lattices

Let $\mathcal{L}$ be a d-dimensional lattice in $\mathbb{R}^m$ and its basis be $\mathbf{B} \in \mathbb{R}^{m \times d}$, we denote $\mathbf{B}_{[i:j)} := (\mathbf{b}_i, \cdots, \mathbf{b}_{j-1})$. We can also denote the lattice generated by basis $\mathbf{B}$ as $\mathcal{L}(\mathbf{B})$. Let $\mathbf{B}^* = (\mathbf{b}_0^*, \ldots, \mathbf{b}_{d-1}^*)$ be the Gram-Schmidt orthogonalization of $\mathbf{B}$, in which $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^*$, $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$, $\forall i \in [d)$. Denote by l_i the logarithm of Gram-Schmidt norm, i.e. $l_i = \ln(\|\mathbf{b}_i^*\|)$, for $i \in [d)$. Let $\mathsf{rr}(\mathbf{B}) = (l_1, \cdots, l_d)$, abbreviate to rr, $\mathsf{rr}_{[i:j)} = (l_i, \cdots, l_{j-1})$. For $i \in [d)$, let π_i be the orthogonal projection over $(\mathbf{b}_0, ..., \mathbf{b}_{i-1})^\perp$. For $0 \le j \le k \le d-1$, we denote by $\mathbf{B}_{\pi[j:k)}$ the local projected block $(\pi_j(\mathbf{b}_j), \pi_j(\mathbf{b}_{j+1}), ..., \pi_j(\mathbf{b}_{k-1}))$, and by $\mathcal{L}_{\pi[j:k)}$ the lattice spanned by $\mathbf{B}_{\pi[j:k)}$. Denote by $\pi_{[j:k)}(\mathbf{v})$ the projection of $\mathbf{v}$ on $\mathrm{span}(\mathcal{L}_{\pi[j:k)})$. Furthermore, if the argument of a projection operator is a set $T \subset \mathbb{R}^d$, it denotes the set containing the projections of all individual vectors within T, i.e., $\pi_*(T) = \{\pi_*(\mathbf{v}) \mid \mathbf{v} \in T\}$. The volume of a lattice $\mathrm{Vol}(\mathcal{L})$ with basis $\mathbf{B}$ is defined as $\mathrm{Vol}(\mathcal{L}) = \sqrt{\mathbf{B}^T\mathbf{B}} = \prod_{i=0}^{d-1} \|\mathbf{b}_i^*\|$. We write $\lambda_i(\mathcal{L})$ for Minkowski's successive minima, the Gaussian Heuristic predicts $\lambda_1(\mathcal{L}) = \mathsf{GH}(\mathcal{L}) := \mathrm{Vol}(\mathcal{L})^{1/d}/\mathrm{Vol}(\mathcal{B}^d(1))^{1/d} \approx \sqrt{\frac{d}{2\pi e}} \cdot \mathrm{Vol}(\mathcal{L})^{1/d}$. We will also denote $\mathsf{GH}(\mathbf{B}) = \mathsf{GH}(\mathsf{rr})$ for a lattice $\mathcal{L}$ with basis $\mathbf{B}$ or profile rr, and $\mathsf{GH}(d)$ for $\mathsf{GH}(\mathcal{L})$ of any d-dimensional lattice $\mathcal{L}$ of volume 1: $\mathsf{GH}(d) \approx \sqrt{\frac{d}{2\pi e}}$.

Definition 2.4 (Discrete Gaussian Distribution). *For a lattice $\mathcal{L} \subset \mathbb{R}^d$, a center vector $\mathbf{c} \in \mathbb{R}^d$, and a parameter $\sigma > 0$, the discrete Gaussian distribution over $\mathcal{L}$ with center $\mathbf{c}$ and parameter σ, denoted by $D_{\mathcal{L},\sigma,\mathbf{c}}$, is defined by the probability mass function:*

$$\mathcal{D}_{\mathcal{L},\sigma,\mathbf{c}}(\mathbf{x}) := \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{x})}{\rho_{\sigma,\mathbf{c}}(\mathcal{L})}, \quad \forall \mathbf{x} \in \mathcal{L}$$

where $\rho_{\sigma,\mathbf{c}}(\mathcal{L}) = \sum_{\mathbf{y} \in \mathcal{L}} \rho_{\sigma,\mathbf{c}}(\mathbf{y})$ is the normalization factor (the total weight over the lattice). Specifically, when $\mathcal{L} = \mathbb{Z}$, we obtain the one-dimensional discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma,c}$ over the integers.

CVP and its Variants [DLvW20]. The analysis of lattice algorithms heavily depends on the Gaussian heuristic, as it better represents the practical complexity of the algorithms than their provable counterparts.

As a direct result a random target $\mathbf{t} \in \mathbb{R}^d$ is expected to lie at distance $\lambda_1(\mathcal{L}) \approx \mathsf{GH}(\mathcal{L})$ from the lattice. This gives the following alternative statements for the common variants of the closest vector problem (CVP).

Definition 2.5 (Closest Vector Problem (CVP)). *Given an arbitrary basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^d$ and a target vector $\mathbf{t} \in \mathrm{span}(\mathcal{L})$. The goal is to find a vector $\mathbf{w} \in \mathcal{L}$ such that $\|\mathbf{w} - \mathbf{t}\| \le \mathsf{GH}(\mathcal{L})$.*

In practice, the concrete hardness of lattice-based cryptography often reduces to specific variants of CVP. To capture these diverse requirements within a unified framework, we introduce the following generalized formulation:

Definition 2.6 ((Batch) Approximate Closest Vector Problem, adaptive version of Def. 2.3 in [PMHS19]). *Let $\mathcal{L} \subset \mathbb{R}^d$ be a lattice with an arbitrary basis $\mathbf{B}$, and $z \in \{1, 2, \infty\}$ denote the ℓ_z norm. Given an approximation factor $\gamma \geq 1$ and a batch of M target vectors $\mathfrak{T} = \{\mathbf{t}_i\}_{i \in [M)} \subset \mathrm{span}(\mathcal{L})$, the goal of the Batch Approximate* CVP *is to find lattice vectors $\{\mathbf{w}_i\}_{i \in [M)} \subset \mathcal{L}$ such that:*

$$\|\mathbf{w}_i - \mathbf{t}_i\|_z \leq \gamma \cdot \mathrm{Dist}^{(z)}(\mathbf{t}_i, \mathcal{L}), \quad \textit{for all } i \in [M).$$

Furthermore, we consider a variant denoted as $\mathsf{CVP}^{(z)}_{D,M}$, where the requirement is $\|\mathbf{w}_i - \mathbf{t}_i\|_z \leq D$ for an absolute distance bound D, independent of the actual distance to the lattice. When $M = 1$, this definition reduces to the single $\mathsf{CVP}^{(z)}_{D}$ under the ℓ_z-norm with an absolute bound D.

In the specific case of the Euclidean norm ($z = 2$), we will assume under the Gaussian Heuristic that $\mathrm{Dist}^{(2)}(\mathbf{t}_i, \mathcal{L}) \approx \mathsf{GH}(\mathcal{L})$ and consider the γ-approximate absolute bound $D = \gamma \cdot \mathsf{GH}(\mathcal{L})$.

In case we are interested to find *all* solutions to the approximate CVP instances, we call this ListCVP, following the list-decoding terminology common in coding-theory. For example, $\mathsf{ListCVP}^{(z)}_{D,M}$ asks to compute for each target vector, all lattice vectors at ℓ_z-distance at most D.

In case $\gamma < 1$ it is unlikely that a solution exists. In this case, we do however consider the problem where such an unlikely solution is promised to exist.

Definition 2.7 (Bounded Distance Decoding Problem (BDD_γ)). *For $0 < \gamma < 1$, an arbitrary basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^d$ and a target vector $\mathbf{t} = \mathbf{w} + \mathbf{e} \in \mathrm{span}(\mathcal{L})$ for which $\mathbf{w} \in \mathcal{L}$ and $\|\mathbf{e}\| \leq \gamma \cdot \mathsf{GH}(\mathcal{L})$. The goal is to recover $\mathbf{w}$.*

By abuse of notation we will typically also write CVP_γ when $\gamma < 1$ to indicate the BDD_γ problem.

For a uniform target $\mathbf{t} \in \mathbb{R}^d/\mathcal{L}$ one can prove that it lies at distance $\gamma \cdot \mathsf{GH}(\mathcal{L})$ from the lattice with probability at most γ^d for $0 < \gamma \leq 1$. For $\gamma \cdot \mathsf{GH}(\mathcal{L}) < \frac{1}{2}\lambda_1(\mathcal{L})$ this estimate is tight. For any $\gamma > 0$ the expected number of lattice points at distance $\gamma \cdot \mathsf{GH}(\mathcal{L})$ for a truly random target is also γ^d. For our later analyses we will Heuristically assume that the targets we encounter satisfy these estimates exactly.

Heuristic 1 (Volume-Based Probability for $\mathsf{BDD/CVP}_\gamma$) *Let $\mathcal{L}$ be a lattice of dimension d and let $\mathbf{t} \in \mathbb{R}^d$ be a random target vector in our procedure. For $0 < \gamma \leq 1$ we assume that*

$$\Pr\Big(\mathrm{Dist}^{(2)}(\mathbf{t}, \mathcal{L}) \leq \gamma \cdot \mathsf{GH}(\mathcal{L})\Big) \;=\; \gamma^d.$$

For $\gamma > 1$, we assume under the Gaussian Heuristic that

$$\sharp\left(\mathcal{L} \cap \mathcal{B}^d(\gamma \cdot \mathsf{GH}(\mathcal{L}))\right) = \gamma^d.$$

For $\gamma \in (0,1)$, this captures the probability that a random target is in fact a BDD_γ instance. The boundary case $\gamma = 1$ corresponds to the generic CVP scenario, in which we assume the target has error satisfying $\|\mathbf{e}\| \approx \mathsf{GH}(\mathcal{L})$, so the worst-case typical error matches the Gaussian Heuristic radius. For $\gamma > 1$ the quantity γ^d transitions to represent the *expected number* of lattice vectors within distance $\gamma \cdot \mathsf{GH}(\mathcal{L})$ from $\mathbf{t}$.

To analyze how the BDD and CVP instances behave in our algorithms we consider the following fairly standard heuristic that the error direction can be modeled as uniform. In particular, this is a heuristic that is already used for the complexity analysis of the (randomized) iterative slicer [DLvW20].

Heuristic 2 (Uniform Error Direction) *Let $\mathcal{L}$ be a d-dimensional lattice and $\mathbf{t} \in \mathcal{L}$ a CVP instance. We assume that the error vector $\mathbf{e} = \mathbf{t} - \mathbf{w}$ to its closest lattice vector $\mathbf{w} \in \mathcal{L}$ has a uniform direction: $\mathbf{e}/\|\mathbf{e}\| \sim \mathcal{S}^{d-1}(1)$.*

Under this heuristic, we can precisely understand the distribution of its projections. This modeling was recently already used to give more accurate estimates in the $\mathsf{BDD}/\mathsf{uSVP}$ setting [dPT26].

Lemma 2.8 (Beta-Distributed Error Projections). *Let $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{d-1})$ be a basis of a d-dimensional lattice $\mathcal{L}$. For a target vector $\mathbf{t}$ and its closest lattice vector $\mathbf{w} \in \mathcal{L}$, assume the error vector $\mathbf{e} = \mathbf{t} - \mathbf{w}$ satisfies Heuristic 2. Then*

$$\frac{\|\pi_{d-\beta}(\mathbf{e})\|^2}{\|\mathbf{e}\|^2} \sim \mathrm{Beta}\left(\frac{\beta}{2}, \frac{d-\beta}{2}\right) \quad \text{for } 0 < \beta < d.$$

2.4 Lattice Reduction

Definition 2.9 (Babai's Nearest Plane Algorithm [Bab86]). *Given a lattice basis $\mathbf{B}$ of $\mathcal{L}$ and a target vector $\mathbf{t} \in \mathbb{R}^d$, Babai's nearest plane algorithm (denoted as* `babai(t, B)`*) outputs the unique lattice vector $\mathbf{w} \in \mathcal{L}$ such that the error vector $\mathbf{e} = \mathbf{t} - \mathbf{w}$ satisfies $\mathbf{e} \in \mathcal{P}(\mathbf{B}^*)$, where $\mathcal{P}(\mathbf{B}^*) = \mathbf{B}^* \cdot [-\frac{1}{2}, \frac{1}{2})^d$ is the centered fundamental parallelepiped of the Gram-Schmidt orthogonalization $\mathbf{B}^*$. Equivalently, the algorithm finds the unique error $\mathbf{e} \in \mathcal{P}(\mathbf{B}^*) \cap (\mathbf{t} + \mathcal{L})$.*

Lemma 2.10 (Klein's Sampler and Norm Bound [Kle00,MR07,GPV08]). *Let $\mathcal{L}$ be a d-dimensional lattice with basis $\mathbf{B}$, $\mathbf{t}$ a target vector, and $0 < \epsilon < 1$ an error parameter. If the standard deviation σ satisfies*

$$\sigma \geq \sqrt{\ln(2d(1+1/\epsilon))/2\pi^2} \cdot \max\|\mathbf{b}_i^*\|,$$

then there exists a procedure `KleinSampler` *which given $\mathbf{B}^*, \sigma, \mathbf{t}$ samples a lattice vector $\mathbf{w} \in \mathcal{L}$ ϵ-close to the discrete Gaussian distribution $\mathcal{D}_{\mathcal{L},\sigma,\mathbf{t}}$ in time $O(d^2)$. Under this condition the output satisfies:*

$$\Pr_{\mathbf{w} \sim \mathcal{D}_{\mathcal{L},\sigma,\mathbf{t}}}\left\{\|\mathbf{w} - \mathbf{t}\| > \sigma\sqrt{2\pi d}\right\} \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-d}. \tag{2}$$

If d is sufficiently large, the tail probability $\frac{1+\epsilon}{1-\epsilon} \cdot 2^{-d}$ becomes negligible in Lemma 2.10. Consequently, the sampler returns a lattice vector $\mathbf{w}$ satisfying the distance bound $\|\mathbf{w}-\mathbf{t}\| \leq \sigma\sqrt{2\pi d}$ with *overwhelming probability*. Setting σ exactly to the condition thus gives a distance bound of

$$\|\mathbf{w}-\mathbf{t}\| \lesssim \sqrt{\frac{d\ln(2d(1+1/\epsilon))}{\pi}} \cdot \max_{i\in[d)} \|\mathbf{b}_i^*\|. \tag{3}$$

Definition 2.11 (HKZ reduction and BKZ reduction [Ngu09]). *The basis $\mathbf{B}$ of a lattice $\mathcal{L}$ is said to be HKZ reduced if $\mathbf{b}_i^* = \lambda_1(\mathcal{L}(\mathbf{B}_{\pi[i:d)}))$, for all $i < d$. It is said BKZ reduced with block-size b(also called as BKZ-b reduced) if $\mathbf{b}_i^* = \lambda_1(\mathcal{L}(\mathbf{B}_{\pi[i:\min\{i+b,d\})}))$, for all $i < d$.*

Heuristic 3 (Root Hermite Factor [CN11]) *To evaluate the quality of a BKZ-b reduced basis $\mathbf{B}$ for a d-dimensional lattice $\mathcal{L}$, the root Hermite factor δ_b is defined such that the length of the first basis vector satisfies*

$$\|\mathbf{b}_0\| = \delta_b^d \cdot \mathrm{Vol}(\mathcal{L})^{1/d}. \tag{4}$$

In the context of the BKZ algorithm with a block size b, empirical results indicate that δ_b is typically bounded within $(1, 1.021]$. For larger block sizes, this factor is heuristically given by the following explicit formula:

$$\delta_b = \left(\frac{b}{2\pi e}(\pi b)^{1/b}\right)^{\frac{1}{2(b-1)}}. \tag{5}$$

Heuristic 4 (Geometric Series Assumption [Sch03]) *For a d-dimensional lattice basis $\mathbf{B}$ that is BKZ-b reduced, the Geometric Series Assumption (GSA) denotes that the norms of Gram-Schmidt vectors after BKZ-b satisfies the condition*

$$\|\mathbf{b}_i^*\| = \alpha_b^i \cdot \|\mathbf{b}_0\|, \quad \text{for } i \in [d).$$

Combined with the definition of the Hermite factor, we get $\alpha_b = \delta_b^{-2d/(d-1)}$.

Lemma 2.12 (D4f-SVP (adapted from [Duc18])). *Let $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{d-1})$ be a basis of a d-dimensional lattice $\mathcal{L}$, and let $\mathbf{s} \in \mathcal{L}$ be the shortest non-zero vector. If the adaptively chosen tail parameter $f \in [d)$ satisfies*

$$0 < \|\pi_f(\mathbf{s})\| \;<\; \sqrt{4/3}\cdot \mathsf{GH}\big(\mathcal{L}(\mathbf{B}_{\pi[f:d)})\big) \quad \text{and} \quad \frac{1}{2}\|\mathbf{b}_i^*\|^2 \;>\; |\langle \mathbf{b}_i^*, \mathbf{s}\rangle|, \quad \forall\, 0 \leq i < f, \tag{6}$$

then the D4f-SVP *procedure, equipped with a sieve database of all short vectors of length at most $\sqrt{4/3}\cdot \mathsf{GH}\big(\mathcal{L}(\mathbf{B}_{\pi[f:d)})\big)$ on the projected sublattice $\mathcal{L}(\mathbf{B}_{\pi[f:d)})$, successfully recovers the shortest vector $\mathbf{s}$.*

The condition on the sieving database is (up to a constant fraction) satisfied by heuristic sieving algorithms. Furthermore, under the GSA, a moderately BKZ-b reduced basis typically satisfies the inner product conditions on the shortest vector $\mathbf{s}$ by a large margin, once the projection condition is satisfied [Duc18]. Heuristically, we thus only have to consider for which f we expect that $\sqrt{4/3}\cdot \mathsf{GH}\big(\mathcal{L}(\mathbf{B}_{\pi[f:d)})\big)$. For $b = \theta(d)$ one obtains under the Gaussian Heuristic and the GSA that $f \approx \frac{d\ln(4/3)}{\ln(d/2\pi e)}$.

2.5 Latest Algorithms for solving CVP_γ with $\gamma = O(1)$

We review two slicer-based primitives used as block oracles in our framework. The first, *Iterative Slicer* (`IterSlicer`, Alg. 1, [SFS09]), is a deterministic procedure whose output quality is controlled by the size of a preprocessed short-vector list. The second, *Randomized Slicer* (`RandSlicer`, Alg. 2, [DLvW20]), augments `IterSlicer` with randomized restarts, enabling target approximation factors below the deterministic threshold at the expense of success probability (and thus repetitions). From Success Probability of Sec. 2.4 in [DLvW20], when a slicer operates on a block of rank d, we write the list size as $|L| = \alpha^d$ and use $\gamma^\star(\alpha) := \frac{\alpha}{2\sqrt{\alpha^2-1}}$ for the upper bound of the approximation factor attained by one run of `IterSlicer`.

The Iterative Slicer. The *Iterative Slicer* (`IterSlicer`, Alg. 1) preprocesses a list $L \subset \mathcal{L} \subset \mathbb{R}^d$ of short lattice vectors, with size parameterized as $|L| = \alpha^d$. Given a target $\mathbf{t}$ (or its projection), `IterSlicer` repeatedly subtracts the closest $\mathbf{w} \in L$ until no further reduction is possible, and outputs $\mathbf{t} - \mathbf{t}'$ as an approximate solution. For a fixed $\alpha > 1$, a single pass of `IterSlicer` heuristically achieves the approximation factor $\gamma^\star(\alpha)$ *with success probability* 1. Larger α (hence larger $|L|$) yields smaller $\gamma^\star(\alpha)$ but higher preprocessing and per-step costs. As a limiting case, if L contained all shortest vectors with size $2^{d/2+o(d)}$, then $\gamma^\star(\alpha)$ and `IterSlicer` returns the exact closest vector.

input : A target vector $\mathbf{t} \in \mathbb{R}^d$, a short vector list $L \subset \mathcal{L}$;
output : The close vector $\mathbf{w} \in \mathcal{L}$;

Function `IterSlicer`$(\mathbf{t}, L)$**:**
  $\mathbf{t}' \leftarrow \mathbf{t}$;
  repeat
    $\mathbf{t}' \leftarrow \mathbf{t}' - \mathbf{w}$, for $\mathbf{w} = \arg\min_{\mathbf{w} \in L} \|\mathbf{t}' - \mathbf{w}\|$;
  until $\|\mathbf{t}' - \mathbf{w}\| \geq \|\mathbf{t}'\|$, $\forall \mathbf{w} \in L$;
  return $\mathbf{t} - \mathbf{t}'$;

Algorithm 1: Iterative Slicer [SFS09]

The Randomized Iterative Slicer. When $\gamma < \gamma^\star(\alpha)$ and the target is randomized by a coset shift, i.e., sampled from $\mathbf{t} + \mathcal{L}(\mathbf{B})$, we use the randomized variant of `IterSlicer`, denoted by `RandSlicerTour` and specified in Alg. 2. One execution of `RandSlicerTour` succeeds with probability $\mathbb{P}_{\alpha^2,\gamma_1^2}(d)$, where the dependence on the dimension d is made explicit. This corresponds to the single-run success probability denoted by $\mathbb{P}_{\alpha^2,\gamma_1^2}$ in Heuristic Claim 1 of [DLvW20].

The *Randomized Iterative Slicer* [DLvW20] (abbreviated as $\texttt{RandSlicer}_{\gamma_1}$, Alg. 2) leverages this probabilistic behavior via randomized restarts. Notably, we modify the original algorithm so that, for any given target vector, it explicitly outputs a required number of valid solutions N_{sol} rather than a single candidate. To guarantee N_{sol} valid solutions per target, the algorithm first calculates a

repetition factor N based on the theoretical success probability $\mathbb{P}_{\alpha^2,\gamma_1^2}(d)$. For each original target $\mathbf{t} \in \mathfrak{T}$, it draws N independent coset representatives $\mathbf{t}' \leftarrow \texttt{Sample}(\mathbf{t} + \mathcal{L}(\mathbf{B}))$, compiling them into a randomized target pool $L_{\mathfrak{T}}$. It then executes `IterSlicer` on every $\mathbf{t}' \in L_{\mathfrak{T}}$ alongside a preprocessed short-vector list $L \subseteq \mathcal{L}(\mathbf{B})$ to generate a set $W := \cup_{\mathbf{t}\in\mathfrak{T}} W_{|\mathbf{t}}$, where $W_{|\mathbf{t}}$ contains the N_{sol} candidate vectors returned by the slicer for the target $\mathbf{t}$. By systematically generating N fresh coset shifts per target, `RandSlicer` effectively amplifies the overall success probability while keeping the target approximation factor γ_1 fixed. Finally, the algorithm filters this set to return the $M \cdot N_{\text{sol}}$ vectors that satisfy the $\mathsf{CVP}^{(2)}_{\gamma_1\cdot\mathsf{GH}(\mathcal{L}),M}$ bounding condition for $\mathfrak{T}$. The precise success probabilities and costs are explained further in Appendix A. For now we will consider the following heuristic to summarize the output of the $\texttt{RandSlicer}_{\gamma_1}$ algorithm.

Heuristic 5 (Randomized Slicer) *Let $\mathbf{B}$ be a d-dimensional lattice basis, let $L \subset \mathcal{L}(\mathbf{B})$ be a list of all α^d shortest vectors, and let $N_{\text{sol}} \geq 1, \gamma_1 > 0$ and $D := \gamma_1 \cdot \mathsf{GH}(\mathcal{L}(\mathbf{B}))$. For a batch $\mathfrak{T} = \{\mathbf{t}_i \in \mathbb{R}^d\}$ of M targets, the $\texttt{RandSlicer}_{\gamma_1}$ procedure recovers for each target N_{sol} solutions of the $\mathsf{CVP}^{(2)}_D$ problem. Furthermore, under Heuristic 1, if $N_{\text{sol}} = \gamma_1^d$, it recovers for each target all γ_1^d $\mathsf{CVP}^{(2)}_D$ solutions.*

For $0 < \gamma_1 \leq 1$, the above heuristic considers BDD_{γ_1} instances and $N_{\text{sol}} = 1$.

3 CVP oracle with dimension for free technique

The dimensions for free procedure [Duc18] (see Lemma 2.12) and the primal attack were originally introduced to reduce the required sieving dimension to solve a (u)SVP problem. This general idea was extended to act directly on BDD instances by [KKNM25], who introduced the Tail-BDD procedure. This procedure, efficiently resolves BDD instances if the error's projection onto the tail subspace is sufficiently small, such that it can be found by a $\mathsf{ListCVP}_{\gamma_1}$ oracle (e.g. instantiated by $\texttt{RandSlicer}_{\gamma_1}$), and allowing Babai's nearest-plane lifting on the remaining dimensions to deterministically recover the target lattice vector. We formalize this mechanism beyond BDD to (approximate) CVP instances via the $\texttt{RandSlicerD4f}_{\gamma_1,\gamma_2,f}$ (Alg. 3).

$\texttt{RandSlicerD4f}_{\gamma_1,\gamma_2,f}$ utilizes the short vector list L, the dimension-for-free value f, and generates N_{sol} solutions from one $\texttt{RandSlicer}_{\gamma_1}$ before dimension-for-free. γ_1 represents the approximation factor of the vectors output directly by the $\texttt{RandSlicer}_{\gamma_1}$ tour, and γ specifies the final approximation factor after the extend-left operation.

The algorithm first calls $\texttt{RandSlicer}_{\gamma_1}$ and generates a set $W = \cup_{\mathbf{t}\in\mathfrak{T}} W_{|\mathbf{t}}$. It then independently lifts each candidate branch using Babai's algorithm and subsequently applies a pruning step, returning only those solutions that satisfy the $\mathsf{CVP}^{(2)}_{\gamma\cdot\mathsf{GH}(\mathcal{L}),M}$ quality bound. Notably, this algorithmic framework is applicable to both BDD and general $\mathsf{ListCVP}_\gamma$ instances, the underlying theoretical bounds and the strategy for selecting the tail parameter f diverge significantly between these two settings.

Parameters : The approximate factor γ_1;
Input : A batch of target vectors $\mathfrak{T} = \{\mathbf{t}_i \in \mathbb{R}^d\}_{i\in[M)}$; the lattice basis $\mathbf{B}$; a short vector list $L \subset \mathcal{L}(\mathbf{B})$ with size $\alpha^{d/2}$; the required amount of solutions for each target N_{sol};
Output : The batch of $\mathsf{CVP}^{(2)}_{\gamma_1\cdot\mathsf{GH}(\mathcal{L}),M}$ solutions for $\mathfrak{T}$;

Function $\texttt{RandSlicer}_{\gamma_1}(\mathfrak{T}, \mathbf{B}, L, N_{\text{sol}})$**:**
  $N \leftarrow N_{\text{sol}} \cdot \max\{\lfloor 1/\mathbb{P}_{\alpha^2,\gamma_1^2}(d)\rceil, 1\}$;
  $W \leftarrow$ `RandSlicerTour`$(\mathfrak{T}, \mathbf{B}, L, N)$;
  return $M \cdot N_{\text{sol}}$ *Vectors from* W *that are* $\mathsf{CVP}^{(2)}_{\gamma_1\cdot\mathsf{GH}(\mathcal{L}),M}$ *solutions for* $\mathfrak{T}$;
Function `RandomizeTargets`$(\mathfrak{T}, \mathbf{B}, N)$**:**
  $L_{\mathfrak{T}} \leftarrow \emptyset$;
  for $\mathbf{t} \in \mathfrak{T}$ **do**
    $L_{\mathbf{t}} \leftarrow \emptyset$;
    while $|L_{\mathbf{t}}| < N$ **do**
      $\mathbf{t}' \leftarrow$ `Sample`$(\mathbf{t} + \mathcal{L}(\mathbf{B}))$;
      $L_{\mathbf{t}} \leftarrow L_{\mathbf{t}} \cup \{\mathbf{t}'\}$;
    $L_{\mathfrak{T}} \leftarrow L_{\mathfrak{T}} \cup L_{\mathbf{t}}$;
  return $L_{\mathfrak{T}}$; // $|L_{\mathfrak{T}}| = N \cdot |\mathfrak{T}|$.
Function `RandSlicerTour`$(\mathfrak{T}, \mathbf{B}, L, N)$**:**
  $L_{\mathfrak{T}} \leftarrow$ `RandomizeTargets`$(\mathfrak{T}, \mathbf{B}, N)$;
  $W = \emptyset$;
  for $\mathbf{t} \in L_{\mathfrak{T}}$ **do**
    $\mathbf{w} \leftarrow$`IterSlicer`$(\mathbf{t}, L)$;
    $W \leftarrow W \cup \{\mathbf{w}\}$;
  return W;

Algorithm 2: Randomized Iterative Slicer [DLdW19]

3.1 Dimensions for free technique in BDD case

For the BDD case, we formulate an adaptive version of the success condition previously defined in [KKNM25], which proposed the Tail-BDD procedure to solve a BDD instance by first finding the projected solution on the tail block $\mathbf{B}_{\pi[f:d)}$ using a bounded CVP solver like $\texttt{RandSlicer}_{\gamma_1}$. It then deterministically recovers the full vector $\mathbf{w}$ by lifting this partial solution through the first f dimensions using Babai's nearest plane algorithm.

Heuristic 6 (Tail-BDD, adapted Claim 3.4 [KKNM25]) *Assume* $(\mathbf{B}, \mathbf{t})$ *is a* BDD *instance with* $\mathbf{t} = \mathbf{w}+\mathbf{e}$*, where* $\mathbf{w}$ *is the solution and* $\mathbf{e}$ *is the corresponding error. Let* $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{d-1})$ *be a BKZ-reduced lattice basis. If the adaptively chosen tail parameter* $f \in [0 : d)$ *satisfies the conditions*

$$\|\pi_f(\mathbf{e})\| < \mathsf{GH}(\mathcal{L}(\mathbf{B}_{\pi[f:d)})) \quad \text{and} \quad \frac{1}{2}\|\mathbf{b}_i^*\|^2 \geq |\langle \mathbf{e}, \mathbf{b}_i^* \rangle|, \quad \forall\ 0 \leq i < f, \tag{7}$$

then under Heuristic 5 the Tail-BDD *procedure successfully recovers the closest vector* $\mathbf{w}$ *with a single call to* $\texttt{RandSlicer}_{\gamma_1}$ *with* $\gamma_1 = 1$.

Parameters : The approximate factor γ_1 for one call of $\texttt{RandSlicer}_{\gamma_1}$, the output approximate factor γ; The d4f value f;
Input : A batch of target vectors $\mathfrak{T} = \{\mathbf{t}_i \in \mathbb{R}^d\}_{i\in[M)}$; the lattice basis $\mathbf{B}$, a short vector list $L \subset \mathcal{L}(\mathbf{B})$ with size $\alpha^{(d-f)/2}$; the required amount of solutions for each target N_{sol};
Output : The batch of $\mathsf{CVP}^{(2)}_{\gamma\cdot\mathsf{GH}(\mathcal{L}),M}$ solutions for $\mathfrak{T}$;

Function $\texttt{RandSlicerD4f}_{\gamma_1,\gamma_2,f}(\mathfrak{T}, \mathbf{B}, L, N_{\text{sol}})$**:**
  $W \leftarrow \texttt{RandSlicer}_{\gamma_1}(\pi_f(\mathfrak{T}), \mathbf{B}_{\pi[f:d)}, L, N_{\text{sol}})$; // $W := \cup_{\mathbf{t}\in\mathfrak{T}} W_{|\mathbf{t}}$
  foreach $\mathbf{t} \in \mathfrak{T}$ **do**
    foreach $\mathbf{w} \in W_{|\mathbf{t}}$ **do**
      $\mathbf{w} \leftarrow \mathbf{w} + \texttt{babai}(\pi_{[f)}(\mathbf{t} - \mathbf{w}), \mathbf{B}_{\pi[f)})$;
      Update $\mathbf{w}$ in $W_{|\mathbf{t}}$;
  return *Vectors from* W *that are* $\mathsf{CVP}^{(2)}_{\gamma\cdot\mathsf{GH}(\mathcal{L}),M}$ *solutions for* $\mathfrak{T}$;

Algorithm 3: Randomized Iterative Slicer with Dimensions for Free.

Following [BN24], we apply Heuristic 2 and Lemma 2.8 with $\beta = d - f$. Then we have $\frac{\|\pi_f(\mathbf{e})\|^2}{\|\mathbf{e}\|^2} \sim \text{Beta}\left(\frac{d-f}{2}, \frac{f}{2}\right)$. Assume $\|\mathbf{e}\| = \gamma_{\mathsf{BDD}} \cdot \mathsf{GH}(\mathcal{L}(\mathbf{B}))$, the success condition (7) of Heuristic 6 is met with probability

$$p(f) \;=\; \Pr\left[\text{Beta}\left(\tfrac{d-f}{2}, \tfrac{f}{2}\right) \le \frac{\mathsf{GH}^2(\mathcal{L}(\mathbf{B}_{\pi[f:d)}))}{\gamma^2_{\mathsf{BDD}} \cdot \mathsf{GH}^2(\mathcal{L}(\mathbf{B}))}\right], \tag{8}$$

so for a chosen failure tolerance $\epsilon > 0$ the largest admissible f is

$$f_{\max} \;=\; \max\{\, f \in [0 : d-1] \;:\; p(f) \ge 1 - \epsilon \,\}. \tag{9}$$

3.2 Dimensions for Free in $\mathsf{ListCVP}_\gamma$ case with $1 \le \gamma \le \sqrt{4/3}$

The original d4f technique [Duc18] was designed to accelerate the search for a single shortest vector by fixing the number of free dimensions at a theoretical maximum. Recently, [WXG25] generalized this concept into a flexible D4F approach. By introducing a tunable dimension parameter, their method enables a precise trade-off between computational time and the size of the output list of short vectors.

Building upon this flexibility, we analyze the d4f value of $\texttt{RandSlicerD4f}_{\gamma_1,\gamma_2,f}$ with $1 \le \gamma_1 \le \sqrt{4/3}$ when using it to solve $\mathsf{ListCVP}_{\gamma_2}$ for $1 \le \gamma_2 \le \gamma_1$. This integration allows us to efficiently generate a batch of candidate approximate closest vectors with a constant approximation $1 \le \gamma \le \gamma_1$ by calling $\texttt{RandSlicer}_{\gamma_1}$ on a projected sublattice $\mathcal{L}(\mathbf{B}_{\pi[f:d)})$. To formally analyze the behavior and success probability of $\texttt{RandSlicerD4f}_{\gamma_1,\gamma_2,f}$, our approach relies on Heuristic 5 and the following lemma.

Lemma 3.1 ($\mathsf{D4f\text{-}CVP}_{\gamma_2}$)**.** *Let* $\mathbf{B}$ *be a* d*-dimensional lattice basis, Let* f *be an adaptively chosen tail parameter* $f \in [0 : d)$*. Then under Heuristic 5 the* D4f-CVP

procedure, with a single call to a $\mathsf{ListCVP}_{\gamma_1}$ *oracle on* $\mathcal{L}(\mathbf{B}_{[f:d)})$*, recovers all solutions to the* $\mathsf{ListCVP}_{\gamma_2}$ *instance* $\mathbf{t}$ *whose errors satisfy*

$$\|\pi_f(\mathbf{e})\| \;\le\; \gamma_1 \cdot \mathrm{GH}\big(\mathcal{L}(\mathbf{B}_{[f:d)})\big) \quad \textit{and} \quad |\langle \mathbf{e}, \mathbf{b}_i^* \rangle| \;\le\; \frac{1}{2}\|\mathbf{b}_i^*\|^2, \quad \forall 0 \le i < f, \tag{10}$$

For the traditional exact case, we set $\gamma_2 = 1$ *and* $\gamma_1 = \sqrt{4/3}$.

Assume $\|\mathbf{e}\| = \gamma_2 \mathsf{GH}(\mathcal{L})$, combining (10) with Heuristic 2 and Lemma 2.8 under the typical-error assumption yields the maximal admissible tail parameter for the second model:

$$f_{\max}^{(2)} \;=\; \max\big\{\, f \in [0:d) \;\big|\; p(f) \ge 1 - \epsilon \,\big\}, \tag{11}$$

where

$$p(f) \;=\; \Pr\left[\mathrm{Beta}\Big(\tfrac{d-f}{2}, \tfrac{f}{2}\Big) \le \left(\frac{\gamma_1 \mathrm{GH}(\mathcal{L}(\mathbf{B}_{\pi[f:d)}))}{\gamma_2 \mathrm{GH}(\mathcal{L}(\mathbf{B}_{[0:d)}))}\right)^2\right].$$

Here ϵ determines the fraction of solutions we would miss. Under this model, the projection is bounded by its expected (squared) length $\gamma_2^2 \cdot \frac{d-f}{d} \cdot \mathsf{GH}(\mathcal{L})^2$ with probability at least $\frac{1}{2} - \Theta(1/\sqrt{f})$. For the typical large values of f, d we consider this is sufficiently close to $\frac{1}{2}$.

Theorem 3.2. *Assume the Geometric Series Assumption (GSA) and the Gaussian Heuristic hold. Consider parameters* $\gamma_2, \gamma_1, b, d, f$ *satisfying* $1 \le \gamma_2 \le \gamma_1 \cdot \delta_b^{-\frac{fd}{d-1}}$ *with* $2 \le b \le d$, $0 \le f < d$. *Let* $\mathcal{L}$ *be a lattice of dimension* d *given by a BKZ-*b *reduced basis* $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{d-1})$, *and let* $(\mathbf{B}, \mathbf{t})$ *be a* CVP_{γ_2} *instance with target* $\mathbf{t}$ *and complete solution set* $W \subset \mathcal{L}$. *Then, under Heuristics 1, 2, and 5,* $\texttt{RandSlicerD4f}_{\gamma_1,\gamma_2,f}$ *outputs the list* W *containing a constant fraction* $\ge \frac{1}{2} - O(1/\sqrt{f})$ *of all* CVP_{γ_2} *solutions of expected size approximately* γ_2^d.

Proof. Let $E = \{\mathbf{e} = \mathbf{t} - \mathbf{w} : \mathbf{w} \in \mathcal{L} \wedge \|\mathbf{e}\| \le \gamma_2 \cdot \mathsf{GH}(\mathcal{L})\}$ for $1 \le \gamma_2 \le \sqrt{4/3} \cdot \delta^{-\frac{fd}{d-1}}$. From Heuristic 1, we have $|E| \approx \gamma_2^d$. Next, we show that for most error vector $\mathbf{e} = \mathbf{t} - \mathbf{w} \in E$, its corresponding projected lattice vector $\pi_f(\mathbf{w})$ is heuristically contained in the candidate list generated by $\texttt{RandSlicer}_{\gamma_1}$ on $\mathcal{L}_{[f:d)}$. Consequently, the projected error $\pi_f(\mathbf{e})$ can be implicitly recovered via the relation $\pi_f(\mathbf{e}) = \pi_f(\mathbf{t}) - \pi_f(\mathbf{w})$. Following Heuristic 2 we can assume that a constant-fraction of at least $\frac{1}{2} - O(1/\sqrt{f})$ of the solution errors satisfies that $\|\pi_f(\mathbf{e})\| \le \sqrt{(d-f)/d} \cdot \|\mathbf{e}\|$.

So under Heuristic 5 and Lemma 3.1 we need to show that $\gamma_2 \cdot \sqrt{\frac{d-f}{d}}\mathsf{GH}(\mathcal{L}) \le \gamma_1 \cdot \mathsf{GH}(\mathcal{L}_{[f:d)})$. Since the input lattice basis is BKZ-b reduced, according to the GSA, the Gram-Schmidt norms follow $\|\mathbf{b}_i^*\| = \delta^d \cdot \alpha^i$. Under the GSA and Heuristic 2 it also follows that the inner product property is true for almost all error. Under the GSA, the Gaussian Heuristic for the full lattice is $\mathsf{GH}(\mathcal{L}) \approx \sqrt{\frac{d}{2\pi e}}\delta^d \alpha^{\frac{d-1}{2}}$. Similarly, for the projected sublattice $\mathcal{L}_{\pi[f:d)}$ of dimension $d-f$, we have $\mathrm{GH}(\mathcal{L}_{[f:d)}) \approx \sqrt{\frac{d-f}{2\pi e}}\delta^d \alpha^{\frac{f+d-1}{2}}$. Substituting these into our initial inequality,

we obtain $\gamma_2 \cdot \sqrt{\frac{d-f}{d}} \cdot \sqrt{\frac{d}{2\pi e}} \delta^d \alpha^{\frac{d-1}{2}} \leq \gamma_1 \cdot \sqrt{\frac{d-f}{2\pi e}} \delta^d \alpha^{\frac{f+d-1}{2}}$. Simplifying the left-hand side by cancelling $\sqrt{d}$ yields $\gamma_2 \cdot \sqrt{\frac{d-f}{2\pi e}} \delta^d \alpha^{\frac{d-1}{2}} \leq \gamma_1 \cdot \sqrt{\frac{d-f}{2\pi e}} \delta^d \alpha^{\frac{f+d-1}{2}}$. By dividing both sides by the common factor $\sqrt{\frac{d-f}{2\pi e}} \delta^d$, we have $\gamma_2 \cdot \alpha^{\frac{d-1}{2}} \leq \gamma_1 \cdot \alpha^{\frac{f+d-1}{2}}$, which cleanly simplifies to: $\gamma_2 \leq \gamma_1 \cdot \alpha^{f/2}$. Given $\alpha = \delta^{-\frac{2d}{d-1}}$ from the GSA, we have $\alpha^{f/2} = \delta^{-\frac{fd}{d-1}}$. Thus, the condition holds naturally, which is formally promised by setting $\gamma_2 \leq \gamma_1 \cdot \delta^{-\frac{fd}{d-1}}$. □

4 Randomized Nearest-Colattice

4.1 The original abstract Nearest-Colattice Framework

The Nearest-Colattice Algorithm was proposed in [EK20, Alg. 2]. It gave a new algorithm to solve $\mathsf{CVP}^{(2)}_{\gamma \cdot \mathsf{GH}(\mathcal{L})}$. Assume there is a d dimensional lattice $\mathcal{L}$ and a target vector $\mathbf{t}$. It firstly constructs a filtration $\{0\} = \mathcal{L}_0 \subset \mathcal{L}_1 \subset \mathcal{L}_2 \subset \cdots \subset \mathcal{L}_k = \mathcal{L}$ of lattice $\mathcal{L}$ with $0 = \mathrm{rk}(\mathcal{L}_0) < \mathrm{rk}(\mathcal{L}_1) < \mathrm{rk}(\mathcal{L}_2) < \cdots < \mathrm{rk}(\mathcal{L}_k) = \mathrm{rk}(\mathcal{L})$, where $\mathcal{L}_{i-1}$ is the sublattice of $\mathcal{L}_i$ for i from 1 to k. Then, it iteratively calls a CVP solver to find the close vector to the projected vector of the target vector $\mathbf{t}$ on a quotient lattice $\mathcal{L}_i/\mathcal{L}_{i-1}$, from $i = k$ to 1.

4.2 Randomized Nearest-Colattice Algorithm

We adapt the theoretical framework in Sec. 4.1 into a more practical version (Alg. 4) by explicitly mapping the abstract operations on the quotient lattice $\mathcal{L}_i/\mathcal{L}_{i-1}$ to numerical operations on orthogonally projected sub-lattice blocks. Operating on a batch of target vectors $\mathfrak{T}$, a lattice basis $\mathbf{B}$, and a set of short vector lists $\mathbb{L} = \{L_0, \ldots, L_{k-1}\}$, the algorithm is governed by a comprehensive stratification strategy $\Psi_{\text{Slicer}} = (\psi_0, \ldots, \psi_{k-1})$. It initializes a residual target set $\mathfrak{T}' = \mathfrak{T}$ alongside the left block boundary $l = d$, and processes this strategy from $i = k$ down to 1. At each iteration, it sets the block boundaries $(l, r) \leftarrow (l - \beta_i, l)$, and parses the specific block configuration tuple $\psi_i = (\gamma_2^{(i)}, \gamma_1^{(i)}, \beta_i, f_i, N_{\text{sol}}^{(i)})$, where $\beta_i = \mathrm{rk}(\mathcal{L}_i) - \mathrm{rk}(\mathcal{L}_{i-1})$ satisfies $\sum_{i=1}^{k} \beta_i = d$.

During the block execution, if $\beta_i = 1$, we regard it as a dimension-for-free operation and call Babai's Nearest Plane Algorithm to find the solution. If $\beta_i > 1$, we call the ACVP oracle $\texttt{RandSlicerD4f}_{\gamma_1^{(i)}, \gamma_2^{(i)}, f_i}$ to solve the problem on the projected sub-lattice $\mathcal{L}(\mathbf{B}_{\pi[l:r)})$ and the projected batch targets $\pi_{[l:r)}(\mathfrak{T}')$, then output the projected solution set W within the approximation $\gamma_2^{(i)}$. Recall that $N_{\text{sol}}^{(i)}$ determines the number of solutions to the internal $\texttt{RandSlicer}_{\gamma_1}$ oracle, and not the size of the solution set W, which with the exception of the first block will for our strategies typically satisfy $|W| \approx |\mathfrak{T}'|$ (one solution per input target). To lift these solutions back to the original lattice, the algorithm recovers the coefficient vector $\mathbf{x}$ for each candidate $\mathbf{w}^{(\pi)} \in W_{|\pi_{[l:r)}(\mathbf{t}')}$ and explicitly updates its associated residual via $\mathbf{t}' \leftarrow \mathbf{t}' - \mathbf{B}_{[l:r)}\mathbf{x}$. After accumulating all newly updated

residuals, it proceeds to the next block. Upon completing all blocks, it outputs the approximate closest vector set $W \leftarrow \{\mathbf{t} - \mathbf{t}' : \mathbf{t} \in \mathfrak{T}, \mathbf{t}' \in \mathfrak{T}' \cap (\mathbf{t} + \mathcal{L})\}$, with the underlying distance bound established in Claim 1.

The algorithm dynamically manages coordinate ranges through tracked variables l and r for transparent filtration handling. These enhancements collectively provide both theoretical clarity and practical implementability.

Input : A batch of target vectors $\mathfrak{T} = \{\mathbf{t}_i \in \mathbb{R}^d\}_{i \in [M)}$, a lattice basis $\mathbf{B}$, a strategy Ψ_{Slicer}, where $\Psi_{\text{Slicer}} = (\psi_0, \cdots, \psi_{k-1})$ contains parameters $\psi_i = (\gamma_2^{(i)}, \gamma_1^{(i)}, \beta_i, f_i, N_{\text{sol}}^{(i)})$ for each block, a set of short vector lists $\mathbb{L}_{[k]}$ for the slicer blocks;

Output : The batch of approximate close vectors W satisfies Claim 1;

Function `RandNColattice`$(\mathfrak{T}, \mathbf{B}, \Psi, \mathbb{L})$**:**

- $\mathfrak{T}' \leftarrow \mathfrak{T}$; $l \leftarrow d$; $r \leftarrow d$; $\mathbf{w} \leftarrow \mathbf{0}$;
- **for** i **from** k **to** 1 **do**
  - $l \leftarrow l - \beta_i$;
  - $W \leftarrow$ `RandSlicerD4f`$_{\gamma_1^{(i)}, \gamma_2^{(i)}, f_i}(\pi_{[l:r)}(\mathfrak{T}'), \mathbf{B}_{\pi[l:r)}, L_i, N_{\text{sol}}^{(i)})$;
  - $\mathfrak{T}_{\text{tmp}} \leftarrow \emptyset$;
  - **foreach** $\mathbf{t}' \in \mathfrak{T}'$ **do**
    - **for** $\mathbf{w}^{(\pi)} \in W_{|\pi_{[l:r)}(\mathbf{t}')}$ **do**
      - $\mathbf{x} \leftarrow \mathbf{w}^{(\pi)}$ w.r.t $\mathbf{B}_{\pi[l:r)}$; $\mathbf{w} \leftarrow \mathbf{B}_{[l:r)}\mathbf{x}$;
      - $\mathbf{t}_{\text{tmp}} \leftarrow \mathbf{t}' - \mathbf{w}$;
      - $\mathfrak{T}_{\text{tmp}} \leftarrow \mathfrak{T}_{\text{tmp}} \cup \{\mathbf{t}_{\text{tmp}}\}$;
  - $\mathfrak{T}' \leftarrow \mathfrak{T}_{\text{tmp}}$; $r \leftarrow l$;
- **return** $\{\mathbf{t} - \mathbf{t}' : \mathbf{t} \in \mathfrak{T}, \mathbf{t}' \in \mathfrak{T}' \cap \mathbf{t} + \mathcal{L}\}$;

Algorithm 4: Randomized Nearest-Colattice

Estimate output Distance of `RandNColattice`. To evaluate norm of error $\mathbf{e}$, we begin by defining a metric that characterizes the expected distance based on the chosen block strategy and the quality of the underlying CVP_{γ_1} solvers.

Note that within a given block strategy, the parameter $\gamma_2^{(i)}$ defines the strict target distance bound for that specific block. The remaining parameters $(\gamma_1^{(i)}, \beta_i, f_i, N_{\text{sol}}^{(i)})$ do not alter this bound, but rather must be selected appropriately to ensure the algorithm actually succeeds in finding vectors that satisfy it.

Definition 4.1 (Distance Bound). *Given the Gram-Schmidt lengths* rr *of a* d*-dimensional lattice basis, and a block strategy* Ψ_{Slicer} *characterized by block sizes* β_i *and distance factors* $\gamma_2^{(i)}$ *for* $i \in [k)$*. Let* $d_i = \sum_{j=0}^{i-1} \beta_j$ *be the cumulative block boundaries (with* $d_0 = 0$*). We define the squared distance bound as:*

$$\text{Dist}^2_{\text{RNC}}(\mathsf{rr}, \Psi_{\text{Slicer}}) := \sum_{i=1}^{k} (\gamma_2^{(i)})^2 \mathsf{GH}^2\big(\mathsf{rr}_{[d_{i-1}:d_i)}\big).$$

Under a fixed strategy Ψ_{Slicer}, Claim 1 guarantees that *if* `RandNColattice` successfully produces outputs, *any* returned vector $\mathbf{w}$ will satisfy this specific distance bound.

Claim 1 *Assume the Gaussian Heuristic holds. Let* rr *be the GS-lengths of the* d*-dimensional input lattice basis. Given a strategy* Ψ_{Slicer}*, the distance between the target vector* $\mathbf{t}$ *and any output vector* $\mathbf{w}$ *successfully produced by* `RandNColattice` *satisfies the following bound:*

$$\|\mathbf{e}\| \leq \text{Dist}_{\text{RNC}}(\mathsf{rr}, \Psi_{\text{Slicer}}). \tag{12}$$

Proof. According to Alg. 4, the i-th block oracle returns a projected lattice vector $\mathbf{w}_i^{(\pi)} = \mathbf{B}_{\pi[d_{i-1}:d_i)}\mathbf{x}_i$ with $\mathbf{x}_i \in \mathbb{Z}^{\beta_i}$. We define its corresponding contribution in the original lattice as the coefficient-preserving lift $\mathbf{w}_i := \mathbf{B}_{[d_{i-1}:d_i)}\mathbf{x}_i$. The final vector is constructed as $\mathbf{w} = \sum_{i=1}^{k} \mathbf{w}_i$. The algorithm processes the target in orthogonal subspaces corresponding to the block indices $[d_{i-1} : d_i)$.

Since the projections into these blocks are mutually orthogonal, we can apply the Pythagorean theorem to decompose the total squared error. The total squared distance is simply the sum of the squared distances in each projected block:

$$\begin{aligned}\|\mathbf{e}\|^2 = \|\mathbf{t} - \mathbf{w}\|^2 &= \sum_{i=0}^{k-1} \left\|\pi_{[d_{i-1}:d_i)}(\mathbf{t}) - \mathbf{w}_i^{(\pi)}\right\|^2 \\ &\leq \sum_{i=0}^{k-1} (\gamma_2^{(i)})^2 \mathsf{GH}^2\big(\mathsf{rr}_{[d_{i-1}:d_i)}\big) = \text{Dist}^2_{\text{RNC}}(\mathsf{rr}, \Psi_{\text{Slicer}}). \square\end{aligned}$$

4.3 Hybrid Randomized Nearest-Colattice

We propose the `HybridRandNColattice` (Alg. 5) algorithm, which alters the processing of the final block by doing a randomized sampling step. This modification allows the algorithm to generate a diverse multitude of vectors belonging to the target vector's coset, effectively injecting the necessary randomness to explore the candidate space and capture multiple valid solutions.

It takes as input a batch of target vectors $\mathfrak{T} = \{\mathbf{t}_i\}_{i\in[M)}$, a basis $\mathbf{B}$, a strategy $\Psi = (\Psi_{\text{Slicer}}, \psi_{\text{Klein}})$ with $\Psi_{\text{Slicer}} = (\psi_0, \cdots, \psi_{k-2})$ containing parameters $\psi_i = (\gamma_2^{(i)}, \gamma_1^{(i)}, \beta_i, f_i, N_{\text{sol}}^{(i)})$ and $\psi_{\text{Klein}} = (\beta_{k-1}, N_{\text{Sample}}, \sigma)$ for the last block, and a family of preprocessed short-vector lists $\mathbb{L}_{[k-1)}$. The output is a set of lattice vectors W satisfying $\|\mathbf{w} - \mathbf{t}\|_z \leq D$ for each $\mathbf{w} \in W_{|\mathbf{t}_i}$, where D is directly determined by (z, Ψ), see Sec. 5 in detail.

The algorithm first generates the samples on the projected coset $\pi_{d-\beta_{k-1}}(\mathbf{t}) + \mathcal{L}(\mathbf{B}_{\pi[d-\beta_{k-1}:d)})$. For each original target $\mathbf{t} \in \mathfrak{T}$, `KleinSampler` (Lemma 2.10) is invoked on the projected instance $\big(\pi_{[d-\beta_{k-1}:d)}(\mathbf{t}), \mathbf{B}_{\pi[d-\beta_{k-1}:d)}\big)$ to sample N_{Sample} independent samples $\mathbf{w}^{(\pi)} \in \mathcal{L}(\mathbf{B}_{\pi[d-\beta_{k-1}:d)})$. Each sample is lifted to $\mathcal{L}(\mathbf{B}_{[d-\beta_{k-1}:d)})$ via $\mathbf{w} = \mathbf{B}_{[d-\beta_{k-1}:d)}\mathbf{x}$, where $\mathbf{x}$ is the coordinate vector of $\mathbf{w}^{(\pi)}$ relative to $\mathbf{B}_{\pi[d-\beta_{k-1}:d)}$, and the residual $\mathbf{t}-\mathbf{w}$ is appended to the expanded batch. Iterating this over all M targets yields $\mathfrak{T}_{\text{exd}}$, a pool of $M \cdot N_{\text{Sample}}$ independently randomized residuals.

Parameters : The required norm z;
Input : A batch of target vectors $\mathfrak{T} = \{\mathbf{t}_i \in \mathbb{R}^d\}_{i\in[M)}$, a lattice basis $\mathbf{B}$, a strategy $\Psi = (\Psi_{\text{Slicer}}, \psi_{\text{Klein}})$, where $\Psi_{\text{Slicer}} = (\psi_0, \cdots, \psi_{k-2})$ contains parameters $\psi_i = (\gamma_2^{(i)}, \gamma_1^{(i)}, \beta_i, f_i, N_{\text{sol}}^{(i)})$ for $0 \le i \le k-2$, and $\psi_{\text{Klein}} = (\beta_{k-1}, N_{\text{Sample}}, \sigma)$, a set of short vector lists $\mathbb{L}_{[k-1)}$ for the slicer blocks;
Output : $\mathsf{CVP}_{D,M}^{(z)}$ solutions for $\mathfrak{T}$;

Function $\texttt{HybridRandNColattice}_z(\mathfrak{T}, \mathbf{B}, \Psi, \mathbb{L}_{[k-1)})$**:**
  $\mathbf{t}' \leftarrow \mathbf{t}$; $\mathfrak{T}_{\text{exd}} \leftarrow \emptyset$;
  while $|\mathfrak{T}_{\text{exd}}| < M \cdot N_{\text{Sample}}$ **do**
    $|\mathfrak{T}_{\text{tmp}}| \leftarrow \emptyset$;
    while $|\mathfrak{T}_{\text{tmp}}| < N_{\text{Sample}}$ **do**
      $\mathbf{w}^{(\pi)} \leftarrow \texttt{KleinSampler}(\pi_{[d-\beta_{k-1}:d)}(\mathbf{t}), \mathbf{B}_{\pi[d-\beta_{k-1}:d)}, \sigma)$;
      $\mathbf{x}^\top \leftarrow \mathbf{w}^{(\pi)}$ w.r.t $\mathbf{B}_{\pi[d-\beta_{k-1}:d)}$; $\mathbf{w} \leftarrow \mathbf{B}_{[d-\beta_{k-1}:d)}\mathbf{x}$;
      $\mathfrak{T}_{\text{tmp}} \leftarrow \mathfrak{T}_{\text{tmp}} \cup \{\mathbf{t} - \mathbf{w}\}$;
    $\mathfrak{T}_{\text{exd}} \leftarrow \mathfrak{T}_{\text{exd}} \cup \mathfrak{T}_{\text{tmp}}$;
  $W \leftarrow \texttt{RandNColattice}(\pi_{[d-\beta_{k-1})}(\mathfrak{T}_{\text{exd}}), \mathbf{B}_{\pi[d-\beta_{k-1})}, \Psi_{[k-1)}, \mathbb{L}_{[k-1)})$;
  $\mathfrak{T}_{\text{exd}} \leftarrow \bigcup_{\mathbf{t}\in\mathfrak{T}_{\text{exd}}} \{\mathbf{t} - \mathbf{w} \mid \mathbf{w} \in W_{|\mathbf{t}}\}$;
  $W \leftarrow \{\mathbf{t} - \mathbf{t}' : \mathbf{t} \in \mathfrak{T}, \mathbf{t}' \in \mathfrak{T}_{\text{exd}} \cap \mathbf{t} + \mathcal{L}\}$;
  return *Vectors from* W *corresponding to* $\mathsf{CVP}_{D,M}^{(z)}$ *solutions for* $\mathfrak{T}$;

Algorithm 5: Hybrid Randomized Nearest-Colattice

The expanded batch is then reduced to a batch of CVP_{γ_1} instances on the inner sublattice of dimension $d - \beta_{k-1}$. The algorithm passes $\pi_{[d-\beta_{k-1})}(\mathfrak{T}_{\text{exd}})$, the prefix strategy $\Psi_{[k-1)}$, and the corresponding short-vector lists $\mathbb{L}_{[k-1)}$ to the randomized oracle `RandNColattice` (Alg. 4). By operating in batch mode, the oracle amortizes the nearest-neighbor search across the shared lists $\mathbb{L}_{[k-1)}$. For each element in $\mathfrak{T}_{\text{exd}}$, it returns a set W of approximate closest vectors that meet the prescribed relaxation factor $\gamma_2^{(i)}$ of the i-th block.

The oracle `RandNColattice` returns a set of projected solution candidates $W_{|\mathbf{t}_{\text{exd}}}$ for each intermediate residual target $\mathbf{t}_{\text{exd}} \in \mathfrak{T}_{\text{exd}}$. To produce full-lattice candidates, these projected solutions must be combined with the earlier outputs of the `KleinSampler`. First, it explicitly updates the extended residual set by subtracting the `RandNColattice` outputs: $\mathfrak{T}_{\text{exd}} \leftarrow \bigcup_{\mathbf{t}_{\text{exd}}\in\mathfrak{T}_{\text{exd}}} \{\mathbf{t}_{\text{exd}} - \mathbf{w} \mid \mathbf{w} \in W_{|\mathbf{t}_{\text{exd}}}\}$. Then, the algorithm reconstructs the final candidate set W by computing the differences $\{\mathbf{t} - \mathbf{t}'\}$ between the original targets $\mathbf{t} \in \mathfrak{T}$ and the fully reduced residuals $\mathbf{t}' \in \mathfrak{T}_{\text{exd}} \cap (\mathbf{t} + \mathcal{L})$. Conceptually, since an intermediate residual was formed by the Klein sampler as $\mathbf{t}_{\text{exd}} = \mathbf{t} - \mathbf{w}_{\text{Klein}}$, the fully reduced residual becomes $\mathbf{t}' = \mathbf{t}_{\text{exd}} - \mathbf{w} = \mathbf{t} - (\mathbf{w}_{\text{Klein}} + \mathbf{w})$. Consequently, the operation $\mathbf{t} - \mathbf{t}'$ inherently yields the full-lattice vector $\mathbf{w}_{\text{Klein}} + \mathbf{w}$. Finally, the algorithm filters and returns only the vectors that strictly satisfy the $\mathsf{CVP}_{D,M}^{(z)}$ condition.

5 Estimate the distance and cost for `HybridRandNColattice`

In this section, we evaluate the computational time costs and the expected lengths of the output vectors concerning both ℓ_2 and ℓ_∞ norms. We conduct this evalua-

tion by investigating the standard Euclidean norm models (including D4f-CVP and Prob-BDD) and the infinity norm models (CVP_{γ_2} and $\mathsf{Flexd4f\text{-}CVP}_{\gamma_2}$).

5.1 4 strategy models based on HybridRandNColattice

To unify the parameters across all models, the stratification strategy is now explicitly defined as $\Psi = (\Psi_{\text{Slicer}}, \psi_{\text{Klein}})$. Here, $\Psi_{\text{Slicer}} = (\psi_0, \ldots, \psi_{k-2})$ contains the parameter tuples $\psi_i = (\gamma_2^{(i)}, \gamma_1^{(i)}, \beta_i, f_i, N_{\text{sol}}^{(i)})$ for the intermediate blocks, and $\psi_{\text{Klein}} = (\beta_{k-1}, N_{\text{Sample}})$ configures the tail block. Let β denote the lattice block size, M be the batch size of target vectors, and N_{Sample} represent the required number of samples for a Gaussian sampler. The approximation factor for the slicer operations is denoted by γ_1.

Building upon the time cost of $T_{\texttt{IterSlicer}}$ and $T_{\texttt{RS}}$ introduced in Appendix A, for a batch of M targets, the time cost of a single block (excluding the last block) in HybridRandNColattice with parameters $(\gamma_2, \gamma_1, \beta, f, N_{\text{sol}})$ and N_{Sample} is given by $T_{\texttt{RS}}(\gamma_1, \beta, M \cdot N_{\text{Sample}} \cdot N_{\text{sol}}) = \frac{M \cdot N_{\text{Sample}} \cdot N_{\text{sol}}}{\mathbb{P}_{\alpha^2, \gamma_1^2}(\beta - f)} T_{\texttt{IterSlicer}}(\beta - f)$. Furthermore, since the $O(\beta^2)$ time cost of dimension-for-free operation and Klein's algorithm are negligible compared to a single call of `IterSlicer`, we omit them from our calculations. Moreover, our strategies are parametrized such that all the middle blocks will produce approximately a single solution for each of the $M \cdot N_{\text{Sample}}$ input targets, which for simplicity we will treat as exactly one. Consequently, the total time cost of HybridRandNColattice can be expressed as $T_{\texttt{RNC}}(\Psi) = \sum_{i=0}^{k-2} T_{\texttt{RS}}(\gamma_1^{(i)}, \beta_i - f_i, M \cdot N_{\text{Sample}} \cdot N_{\text{sol}}^{(i)})$.

To specify the block sizes β_i for the strategy Ψ, we partition the lattice dimension based on a target base size β. For all models except Prob-BDD, the lattice is partitioned uniformly by setting $\beta_i = \beta$ for $0 \le i \le k-2$, which leaves a residual tail block of size $\beta_{k-1} = d \bmod \beta$. Conversely, in the Prob-BDD model, the first block is set to $\beta_0 = \beta$, followed by intermediate blocks of size $\beta_i = \beta - f$ for $1 \le i \le k-2$, resulting in a tail block of size $\beta_{k-1} = (d - \beta) \bmod (\beta - f)$. Crucially, if this tail dimension is too small to provide adequate randomness for the sampling phase, we dynamically merge it with the penultimate block. For models utilizing the dimension-for-free technique, the specific magnitude of f is derived from the maximal admissible tail parameter formulated in Eq. (11). The value of f is then maximized such that the probability $p(f)$ remains bounded by the $1 - \epsilon$ threshold, yielding a highly optimized but secure free dimension. Building upon this dimensional foundation, we categorize the configurations into five specific models based on the target norm and application, as detailed in Table 1.

In Table 2, we list the distance bound D of the output vectors produced by the HybridRandNColattice. Specifically, we analyze the recursive projection of the target batch dictated by the stratification strategy, evaluating each block's exact contribution to the global approximation factor. We formally establish that the resulting output distance bound D is strictly parameterized by the tuple (z, Ψ).

Table 1: Configuration Parameters for Different Models.

Norm	Model	First Block (ψ_0)	Middle Block (ψ_i)	Tail Block (ψ_{Klein})
ℓ_2	D4f-CVP	$(1, \sqrt{4/3}, \beta, f, (\frac{4}{3})^{(\beta-f)/2})$	$(1, \sqrt{4/3}, \beta, f, (\frac{4}{3})^{(\beta-f)/2})$	$(\beta_{k-1}, 1)$
	Prob-BDD	$(\gamma_{\text{BDD}}, 1, \beta, f, 1)$	$(1, 1, \beta-f, 0, 1)$	$(\beta_{k-1}, \gamma_{\text{BDD}}^{-\beta})$
ℓ_∞	CVP_{γ_2}	$(\gamma_2, \gamma_2, \beta, 0, 1)$	$(\gamma_2, \gamma_2, \beta, 0, 1)$	$(\beta_{k-1}, \frac{1}{p_\infty})$
	$\text{Flexd4f-CVP}_{\gamma_2}$	$(\gamma_2, \sqrt{4/3}, \beta, f, (\frac{4}{3})^{(\beta-f)/2})$	$(1, \sqrt{4/3}, \beta, f, (\frac{4}{3})^{(\beta-f)/2})$	$(\beta_{k-1}, \frac{1}{p_\infty})$

Table 2: Time Cost and Output ℓ_2 Distance u_i under Different Strategy Models.

Norm	Model	First Block ($i = 0$)		Mid Block ($1 \le i \le k-2$)	
		Time	ℓ_2 Dist.	Time	ℓ_2 Dist.
ℓ_2	D4f-CVP	$T_{\text{RS}}(\sqrt{\frac{4}{3}}, \beta-f, M\cdot(\frac{4}{3})^{(\beta-f)/2})$	$\mathsf{GH}(\mathbf{B}_{\pi[\beta)})$	$T_{\text{RS}}(\sqrt{\frac{4}{3}}, \beta-f, M\cdot(\frac{4}{3})^{(\beta-f)/2})$	$\mathsf{GH}(\mathbf{B}_{\pi[i\beta:(i+1)\beta)})$
	Prob-BDD	$T_{\text{RS}}(1, \beta-f, M\cdot\gamma_{\text{BDD}}^{-(\beta-f)})$	$\gamma_{\text{BDD}}\mathsf{GH}(\mathbf{B}_{\pi[\beta)})$	$T_{\text{RS}}(1, \beta-f, M\cdot\gamma_{\text{BDD}}^{-(\beta-f)})$	$\mathsf{GH}(\mathbf{B}_{\pi[i\beta-(i-1)f:(i+1)\beta-if)})$
ℓ_∞	CVP_{γ_2}	$T_{\text{RS}}(\gamma_2, \beta, M)$	$\gamma_2\mathsf{GH}(\mathbf{B}_{\pi[\beta)})$	$T_{\text{RS}}(\gamma_2, \beta, M)$	$\gamma_2\mathsf{GH}(\mathbf{B}_{\pi[i\beta:(i+1)\beta)})$
	$\text{Flexd4f-CVP}_{\gamma_2}$	$T_{\text{RS}}(\sqrt{\frac{4}{3}}, \beta-f, M\cdot(\frac{4}{3})^{(\beta-f)/2})$	$\gamma_2\mathsf{GH}(\mathbf{B}_{\pi[\beta)})$	$T_{\text{RS}}(\sqrt{\frac{4}{3}}, \beta-f, M\cdot(\frac{4}{3})^{(\beta-f)/2})$	$\mathsf{GH}(\mathbf{B}_{\pi[i\beta:(i+1)\beta)})$

5.2 Estimate Distance for ℓ_2 norm.

In the `HybridRandNColattice` algorithm, the `RandSlicerD4f` is replaced by the `KleinSampler` in the last block to introduce necessary randomness. From Eq. (2) and Claim 1, the squared ℓ_2 norm upper bound of the output distance for `HybridRandNColattice` is initially given by:

$$\text{Dist}^2_{\text{HRNC}}(\mathsf{rr}, \Psi) := \text{Dist}^2_{\text{RNC}}(\mathsf{rr}, \Psi_{\text{Slicer}}) + u^2_{k-1}, \tag{13}$$

where $u_i = \gamma_2^{(i)}\mathsf{GH}(\mathsf{rr}_{[d_{i-1}:d_i)})$ for $0 \le i \le k-2$ and u_{k-1} is the output distance of the `KleinSampler`. As proven in Lemma 5.1, under the Geometric Series Assumption (GSA), the norm of the randomized error injected at the final block is actually negligible compared to the deterministic error produced by the first block.

Negligibility of the Tail-Block Norm. According to the theoretical behavior of the `KleinSampler` established in Eq. (3), it follows that the maximum distance of $\|\pi_{[d_{k-2}:d)}(\mathbf{e})\|$ over the projected tail sublattice $\mathcal{L}(\mathbf{B}_{\pi[d_{k-2}:d)})$ is bounded by

$$u_{k-1} = \sqrt{\frac{\beta_{k-1}\ln(2\beta_{k-1}(1+1/\epsilon))}{\pi}} \cdot \max_{i\in[d_{k-2}:d)} \|\mathbf{b}_i^*\|. \tag{14}$$

Under the GSA, u_{k-1} is actually negligible compared to the deterministic error produced by the first block. To rigorously quantify this observation, we establish the following lemma.

Lemma 5.1. *Let* $\mathbf{B}$ *be a BKZ-b reduced lattice basis with dimension* d *satisfying the GSA (Heuristic 4). Under a uniform block strategy of size* β *(*$d = k\beta$*), let* u_0 *be the* CVP_{γ_2} *distance bound for the first block and* u_{k-1} *be the expected residual distance sampled from the final block. Assume* $k \geq 3$, $\gamma_2 \geq 1$, $\epsilon \in [2^{-10}, 1]$, $b \geq 50$ *and* $\beta \geq 50$, $u_{k-1} \ll u_0$.

Proof. According to the Gaussian Heuristic, GSA and the root Hermite factor(rhf), we estimate upper bound of the CVP_{γ_2} solution output by the first blocks as $\|\mathbf{b}_0^*\| = \gamma_2 \mathsf{GH}(\mathbf{B}_{[\beta]}) = \gamma_2 \sqrt{\frac{\beta}{2\pi e}} \cdot \alpha_b^{\frac{\beta-1}{2}} \|\mathbf{b}_0\|$.

From GSA, we have $\max_{i\in[(k-1)\cdot\beta:d)} \|\mathbf{b}_i^*\| = \|\mathbf{b}_{(k-1)\cdot\beta}^*\| = \alpha_b^{(k-1)\cdot\beta}\|\mathbf{b}_0\|$. For the final block that employs the `KleinSampler`. Based on Eq. (3), we have $u_{k-1} = \sqrt{\frac{\beta \ln(2\beta(1+1/\epsilon))}{\pi}} \cdot \alpha_b^{(k-1)\cdot\beta}\|\mathbf{b}_0\|$.

Then, we have

$$\frac{u_{k-1}}{u_0} = \frac{\sqrt{\frac{\beta \ln(2\beta(1+1/\epsilon))}{\pi}} \cdot \alpha_b^{(k-1)\cdot\beta}\|\mathbf{b}_0\|}{\gamma_2 \sqrt{\frac{\beta}{2\pi e}} \cdot \alpha_b^{\frac{\beta-1}{2}} \|\mathbf{b}_0\|} = \frac{\sqrt{2e \ln(2\beta(1+1/\epsilon))}}{\gamma_2} \cdot \alpha_b^{(k-1)\beta - \frac{\beta-1}{2}}.$$

Assume $k \geq 3$, $\gamma_1 \geq 1$, $\epsilon \geq 2^{-10}$, $b \geq 50$ and $\beta \geq 50$, the ratio $\frac{u_{k-1}}{u_0} \leq 1.76 \times 10^{-4} \ll 1$. □

Following Lemma 5.1, we can omit the tail block's contribution from the overall output distance without compromising the precision of our bounds, yielding the following claim.

Claim 2 *Under the GSA, the squared upper bound of the* `HybridRandNColattice` *output distance can be evaluated as*

$$\mathrm{Dist}_{\mathrm{HRNC}}(\mathsf{rr}, \Psi) \approx \mathrm{Dist}_{\mathrm{RNC}}(\mathsf{rr}, \Psi_{\mathrm{Slicer}}). \tag{15}$$

The norm bound of the intermediate blocks. For the intermediate blocks $(1 \leq i \leq k-2)$, the norm bound is generalized as $u_i = \gamma_2^{(i)} \mathsf{GH}(\mathcal{L}(\mathbf{B}_{\pi[d_i:d_{i+1}]}))$, where $\gamma_2^{(i)}$ is determined by the specific configuration of the invoked oracle. Based on the dimension-for-free parameter f_i in the strategy tuple ψ_i, the oracle execution fundamentally falls into two cases for the above 3 Euclidean norm models:

1. For $f_i = 0$. The oracle `RandSlicerD4f` degenerates into a standard $\texttt{RandSlicer}_{\gamma_1}$ tour. In this scenario, the approximation factor of the output perfectly aligns with the input factor. i.e. $\gamma_2^{(i)} = \gamma_1^{(i)}$.
2. For $f_i > 0$. Following Lemma 3.1 and Heuristic 5, the oracle sets $\gamma_1^{(i)} = \sqrt{4/3}$ and recovers the exact CVP vector $\gamma_2^{(i)} = 1$ after the extend-left operation by filtering through roughly $N_{\mathrm{sol}}^{(i)} = (4/3)^{\beta_i/2}$ lifted candidates.

The norm bound of the first block. For the other two models, the expected norm bound of the first block is identical to that of the intermediate blocks. Therefore, this subsection exclusively focuses on the Prob-BDD model. The idea behind this strategy is that the majority of the distance contribution of the solution comes from the first block. Moreover, if we obtain many targets ($N_{\text{sample}} \gg 1$) in the first block, then we can expect that at least one is exceptionally close to the lattice in the first block. Concretely, by Heuristic 1, any projected error $\tilde{\mathbf{e}}_i = \pi_{[\beta)}(\mathbf{e}_i)$ on the first-block sublattice satisfies

$$\Pr\big(\|\tilde{\mathbf{e}}_i\| \le \gamma_{\mathsf{BDD}} \cdot \mathsf{GH}(\mathcal{L}(\mathbf{B}_{[\beta)}))\big) = \gamma_{\mathsf{BDD}}^{\beta}, \quad \gamma_{\mathsf{BDD}} \in (0,1).$$

So if we set $N_{\text{sample}} = \gamma_{\mathsf{BDD}}^{-\beta}$ we can expect one such exceptionally small error to exist and the corresponding bound on the full first block becomes

$$\|\pi_{[\beta)}(\mathbf{e})\| \;\le\; \gamma_{\mathsf{BDD}} \cdot \mathsf{GH}\left(\mathcal{L}(\mathbf{B}_{[\beta)})\right) = N_{\text{sample}}^{-1/\beta} \cdot \mathsf{GH}\left(\mathcal{L}(\mathbf{B}_{[\beta)})\right).$$

5.3 Estimate Distance for ℓ_∞ norm

Many lattice-based signature schemes, with Dilithium [BDK+21] the most prominent, use the infinity norm for distance bounding. In this part, we analyze the success probability of solving $\mathsf{CVP}_{D,M}^{(\infty)}$ under the ℓ_∞ norm for `HybridRandNColattice`. Just as other existing approaches, this follows a filtering method: we search for candidate vectors that are close in the ℓ_2 norm and aim to find one that also has a small ℓ_∞ distance. This also explains why the Prob-BDD approach is not relevant for the infinity norm: it would simply consider a strict subset of the candidate vectors and therefore have a (typically strictly) lower success-probability.

Suppose we aim to find a vector such that $\|\mathbf{t} - \mathbf{w}\|_\infty \le D$ and the distance of the output vector $\mathrm{Dist}_{\mathrm{RNC}}(\mathsf{rr}, \Psi)$ satisfies Eq. (15).

The error values that occur as solutions of `HybridRandNColattice` belong to the following volume

$$V := \bigotimes_{i=0}^{k-1} \{\mathbf{x} \in \mathrm{span}(\mathbf{B}_{\pi[d_{i-1}:d_i)}) : \|\mathbf{x}\| \le u_i\}. \tag{16}$$

By the Gaussian Heuristic the success probability for such a single error to satisfy $\|\mathbf{e}\|_\infty$ is given by the volume ratio $p_\infty = \mathrm{Vol}(\mathcal{L} \cap V \cap [-D, D]^d)/\mathrm{Vol}(\mathcal{L} \cap V)$. Since the relevant region may be misaligned with the coordinate axes, an exact evaluation of $\mathrm{Vol}(\mathcal{L} \cap V \cap [-D, D]^d)$ is nontrivial. Even numerical methods become quickly infeasible if we consider the common regime for us where the success probability is exponentially small. We therefore adopt the simplified success-probability models under the ℓ_∞ norm from prior works [BDK+21, WXG25], where the error vectors are modeled as uniform and independent in a ball of radius r, where r matches the Euclidean distance bound of our `HybridRandNColattice` procedure.

Definition 5.2. *Given a lattice $\mathcal{L}$. For a vector $\mathbf{v}$ sampled uniformly at random from $\mathcal{L} \cap \mathcal{B}^d(r)$, we define $p_\infty(\mathcal{L}, r, D, d)$ as the probability that its infinity norm is bounded by D (i.e., $\|\mathbf{v}\|_\infty \leq D$):*

$$p_\infty(\mathcal{L}, r, D, d) := \frac{\sharp(\mathcal{L} \cap \mathcal{B}^d(r) \cap [-D, D]^d)}{\sharp(\mathcal{L} \cap \mathcal{B}^d(r))}, \tag{17}$$

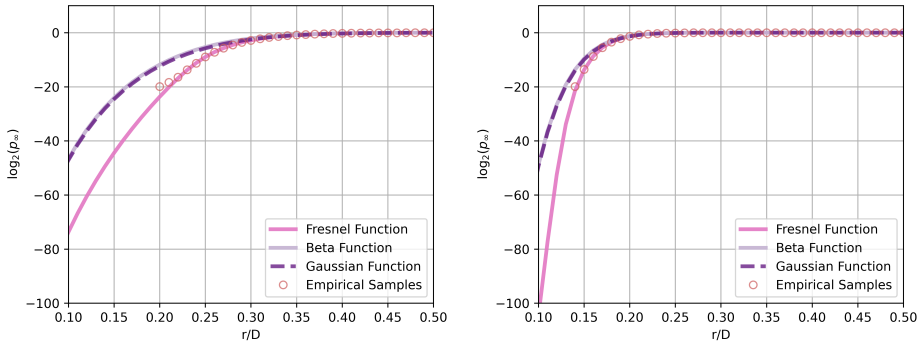
where $\sharp A$ means the amount of lattice points in the set A.

Following this simplified model and the Gaussian Heuristic, we thus estimate the ℓ_∞ acceptance probability as $p_\infty := \Pr(\mathcal{L}, r, d, D)$ (see Definition 5.2) Contrary to the prior case, we can efficiently evaluate this volume ratio in practice. Concretely, given the radius r, bound D, and dimension d, the probability $p_\infty = \Pr(\mathcal{L}, r, d, D)$ is evaluated in three cases: if $r \leq D$, $\Pr(\mathcal{L}, r, d, D) = 1$ obviously; if $r \geq \sqrt{d}D$, the cube $[-D, D]^d$ is completely inside the ball $\mathcal{B}^d(r)$; if $D < r < \sqrt{d}D$, we compute the ball-cube intersection probability using the Fresnel integrals method [DEP23]. As shown in Fig. 1, this integral approximation proves to be the most accurate compared to alternative heuristic approximations based on treating each coefficient of a uniform ball sample as independent Gaussian or their square as a Beta random variables [WXG25, vW23]. As the Fresnel method can be somewhat costly to compute we do when applicable fall back to the Gaussian estimate $p_\infty(\mathcal{L}, r, D, d) \approx \left(1 - 2\Phi\left(-\frac{D\sqrt{d}}{r}\right)\right)^d$, which is accurate in the low-probability regime we target.

While this simplified success probability can be used to fairly compare our method with other methods, we want to emphasize that it does not match the actual ℓ_∞-norm success probability. Our method inherently introduces a bias when sampling multiple solutions: the solution errors are longer in the direction of the first blocks, and shorter in those of the later blocks. Indeed, our practical experiments fail to reach the precise target ℓ_∞-norm. However, the bias is lower than in prior methods based on Kannan's method, such as those that occur in prior cryptanalysis on Dilithium [BDK+21, WXG25]. For these cases, all solutions live in a single sublattice (coset), and in addition it is not even clear that these SVP solutions from a sublattice correspond to a CVP solution at all. In contrast, in our approach each candidate target is randomized already in the last block, and thus the solutions are sufficiently independent and not constrained to any low-rank sublattice (coset).

6 Estimator based on Randomized Nearest-Colattice

Sec. 5 introduced 4 `HybridRandNColattice` strategy models. In this section, we turn these models into a concrete estimator. In Sec. 6.1, we propose an approximate CVP estimator based on the `HybridRandNColattice` approach in two-step mode, describe the common search space and makes explicit which parameters are enumerated. Sec. 6.2 applies the resulting estimator to the security analysis of Dilithium, HuFu, and one-more-ISIS based on three computational cost models (the theoretical model, the classical AGPS20 model [AGPS20], and the classical MATZOV22 model [MAT22]) introduced in Appendix A.



(a) $d = 50$. (b) $d = 200$.

Fig. 1: $\log_2 p_\infty$ with the growth of r/D: (1) "Expirical Samples": the experimental value of $\log_2 p_\infty$; (2) "Fresnel/Beta/Gaussian Function", is the probability estimation by the Fresnel [DEP23]/Beta [vW23]/Gaussian [WXG25] function.

6.1 Approximate CVP Estimator

We distinguish between a `HybridRandNColattice` strategy model and a computational cost model. For the Euclidean norm, the applicable strategy models are D4f-CVP, and Prob-BDD. For the infinity norm, the applicable models are CVP_{γ_2} and $\mathsf{Flexd4f\text{-}CVP}_{\gamma_2}$. Independently, the computational cost is instantiated using one of the theoretical, AGPS20, or MATZOV22 cost models. A cost model changes the cost assigned to a fixed strategy, but not its estimated output distance or success probability.

Given a lattice volume V, dimension d, the norm upper bound D, the batch size of the targets M and the norm type z. The estimator enumerates a progressive BKZ preprocessing with maximum block size b and a value β for setting the blocksize of `RandNColattice` by inputting it into different models. The Prob-BDD model additionally enumerates $\gamma_{\mathsf{BDD}} \in (0, 1)$, whereas the two infinity-norm models enumerate an admissible value of $\gamma_2 \in [1, \sqrt{4/3}]$. No other strategy parameter is searched independently.

For each value of b, the estimator derives a Gram–Schmidt profile rr to model progressive BKZ reduction. It achieves this by first initializing the profile under the Geometric Series Assumption (GSA) with volume V, dimension d, and block size 2, and subsequently evolving it via a BKZ simulator [CN11] from blocksize 3 to maximum block size b. For every value of β and each model-specific approximation parameter, a strategy constructor instantiates the corresponding configuration detailed in Table 1. In particular, parameters such as the number of blocks k, the block dimensions β_i, the output factors $\gamma_2^{(i)}$, the oracle factors $\gamma_1^{(i)}$, the free dimensions f_i, and the candidate multiplicities $N_{\mathrm{sol}}^{(i)}$ are all derived accordingly.

In models that do not use dimensions for free, we set $f_i = 0$. In the Prob-BDD model, the first-block value of f is chosen as the largest value satisfying the

Tail-BDD success condition in Eq. (9). In the D4f-CVP and Flexd4f-CVP$_{\gamma_2}$ models, f_i is chosen following the Eq. (11).[6] The block partition, including the possible merging of a small tail block, is then derived according to the rules of Sec. 5.1.

For the resulting strategy Ψ, the estimator first computes its Euclidean output-radius estimate $r = \mathrm{Dist}_{\mathrm{HRNC}}(\mathsf{rr}, \Psi)$. For $z = 2$, the candidate is retained only when $r \leq D$. For $z = \infty$, the estimator computes p_∞ using the ball–cube intersection model from Sec. 5.3 and sets $N_{\mathrm{Sample}} = \left\lceil \frac{1}{p_\infty} \right\rceil$. For Prob-BDD, the corresponding sampling multiplicity is $N_{\mathrm{Sample}} = \left\lceil \gamma_{\mathsf{BDD}}^{-\beta_0} \right\rceil$, while $N_{\mathrm{Sample}} = 1$ for the D4f-CVP model. We decompose the cost of every candidate into three parts. Let $T_{\texttt{ProBKZ}}$ denote the cost of a progressive BKZ-b reduction, $T_{\texttt{Sieve}}$ the one-time cost of constructing the short-vector lists required by the $\texttt{RandSlicer}_{\gamma_1}$ blocks, and $T_{\texttt{RNC}}$ the target-dependent cost of all $\texttt{RandSlicer}_{\gamma_1}$ in RandNColattice. Each short-vector list is charged once for its corresponding block and is subsequently reused across all target queries and randomized repetitions.

We define our cost estimations under three different models. For the theoretical cost model, denoted as `classical_naive`, the total cost is measured in $\log_2$(bits) and given by $T = \max\{T_{\texttt{ProBKZ}}, T_{\texttt{Sieve}} + T_{\texttt{RNC}}\}$. Alternatively, when considering the AGPS20 and MATZOV22 gate-count models (denoted as `classical_agps20` and `classical_matzov22`, respectively), the total cost is measured in $\log_2$(gates) and computed as $T = T_{\texttt{ProBKZ}} + T_{\texttt{Sieve}} + T_{\texttt{RNC}}$.The progressive BKZ cost is estimated by summing the sieving-based SVP oracle costs over all BKZ tours:$T_{\texttt{ProBKZ}}(b, d) = \sum_{j=2}^{b}(d - j + 1)T_{\texttt{Sieve}}(j)$.When accounting for the dimensions-for-free technique, this calculation is refined to:$T_{\texttt{ProBKZ}}(b, d) = \sum_{j=2}^{b}(d - j + 1)T_{\texttt{Sieve}}(j - \texttt{d4f}(j))$. Detailed formulations for the individual phase costs are deferred to Appendix A. For a fixed strategy model and cost model, the estimator minimizes T over all enumerated parameters. When a single estimate is required for a given norm, we additionally minimize over all applicable strategy models. Besides the minimum cost, the estimator records the minimizing values $(b^\star, \Psi^\star)$, which are reported together with the security estimate.

6.2 Security Estimation for Post-Quantum Schemes

In this section, we evaluate the post-quantum signature schemes Dilithium, one-more-ISIS, and HuFu against forgery attacks using the estimator and time-cost model introduced in Sec. 6. The security of all three schemes under such attacks relies on the hardness of the ISIS problem or its variants. Therefore, to facilitate the subsequent security analysis, we first introduce a reduction from the ISIS problem to an approximate CVP problem. This reduction allows us to leverage CVP-based attack models for estimating the concrete security of these schemes. According to the NIST PQC security criteria [Nat23], the required security levels I–V correspond to 143, 146, 207, 210, and 272 classical gate counts, respectively.

We define the ISIS problem as follows:

[6] In the base case that $\gamma_1^{(i)} = \sqrt{4/3}$, we heuristically simplified under the GSA as $f_i = \frac{\beta_i \ln(4/3)}{\ln(\beta_i/2\pi e)}$ to matching prior work.

Definition 6.1 (ISIS). *Let* $\mathbf{A} \leftarrow_{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{t} \leftarrow_{\$} \mathbb{Z}_q^n$, *the goal of solving* $ISIS_{n,m,B}$ *is to find short vectors* $\mathbf{z} \in \mathbb{Z}_q^m$, $\mathbf{u} \in \mathbb{Z}_q^n$ *such that* $\mathbf{Az} + \mathbf{u} = \mathbf{t} \bmod q$ *and* $\left\| \begin{pmatrix} \mathbf{z} \\ \mathbf{u} \end{pmatrix} \right\| \leq B$.

The reduction from ISIS to approximate CVP has been discussed in [WXG25]. Here, we briefly review it. Let the lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ with column-wise basis $\mathbf{B} = \begin{pmatrix} q\mathbf{I} & \mathbf{A} \\ \mathbf{O} & \mathbf{I} \end{pmatrix}$ and the target vector of the approximate CVP is $\begin{pmatrix} \mathbf{t} \\ \mathbf{0} \end{pmatrix}$ and we want to find the approximate close vector $\begin{pmatrix} \mathbf{Az} \\ \mathbf{z} \end{pmatrix} \in \mathcal{L}(\mathbf{B})$.

Inaccuracies in the Estimation. While in the following sections we will present concrete bit-estimates coming from our estimator, we want to make clear that these numbers should not be taken as a fully accurate claim of the precise attack cost, nor as a completely fair comparison with e.g. the Restricted SVP [WXG25] approach (in either direction). Instead, they are meant to demonstrate which strategies perform best and to validate that the algorithms are competitive with the best practical Kannan's embedding approach, while not relying on the possibly faulty heuristic that the highly biased SVP solutions indeed correspond to CVP solutions. We furthermore make the inaccuracies explicit:

- As already mentioned, for the Restricted SVP [WXG25] estimate, we used the optimistic probability of 0.343 to estimate the fraction of SVP solutions that correspond to a CVP solution. In practice, this probability could in fact go to 0, which is the main inaccuracy our method resolves: the solutions of `HybridRandNColattice` are guaranteed CVP solutions.
- For $\texttt{RandSlicer}_{\gamma_1}$ we used the asymptotic success probability formulas $\mathbb{P}_{\alpha^2,\gamma_1^2}(d)$ from [DLvW20], ignoring polynomial factors, as it is still an open problem to determine it exactly. Experiments from [KKNM25] indicate that these factors are reasonably small in practice. Our own experiments in Section 7 for the ℓ_2-norm seem to validate this: our implementation indeed recovers the target solutions when parametrized with these formulas.
- For both estimators, and all existing prior works, we used an optimistic model for the ∞-norm success probability. Our experiments in Section 7 indicate that indeed our algorithm does not recover the target ℓ_∞-norm when parametrized following this model. We believe however that the bias in our method (as explained later) could be lower than in [WXG25], and we make the actual probability explicit. We present the computation of the exact probability as an explicit open problem in Section 8.

Security Estimation for Dilithium. In Dilithium, its signature forgery attack relies on the hardness of the $\text{ISIS}^{(\infty)}$ problem, which is the approximate CVP considering the infinity norm. We define the $\text{ISIS}^{(\infty)}$ problem as follows:

Definition 6.2 (ISIS$^{(\infty)}$). *Let* $\mathbf{A} \leftarrow_{\$} \mathbb{Z}_q^{m\times n}$, $\mathbf{t} \leftarrow_{\$} \mathbb{Z}_q^n$, *the goal of solving* $ISIS^{(\infty)}_{n,m,B}$ *is to find short vectors* $\mathbf{z} \in \mathbb{Z}_q^m$, $\mathbf{u} \in \mathbb{Z}_q^n$, $\|\mathbf{z}\|_\infty \le B_{\mathbf{z}}$, $\|\mathbf{u}\|_\infty \le B_{\mathbf{u}}$ *such that* $\mathbf{Az} + \mathbf{u} = \mathbf{t} \bmod q$.

The Dilithium specification adopts a unified bound $B = \max\{B_{\mathbf{u}}, B_{\mathbf{z}}\}$, where $B_{\mathbf{u}}$ and $B_{\mathbf{z}}$ are defined in Equation (7) of the Dilithium paper [BDK+21]. To ensure consistency with this specification, our analysis in this paper adopts the same approach.

Table 3: Security Estimation for Dilithium.

Estimator	`classical_agps20`			`classical_matzov22`		
	Lv2	Lv3	Lv5	Lv2	Lv3	Lv5
Required Level	146	207	272	146	207	272
Case 1: $B = \max\{B_{\mathbf{u}}, B_{\mathbf{z}}\}$						
Value of B	350209	724481	769537	350209	724481	769537
Restricted SVP	**159.9**	**223.0**	**300.3**	-	-	-
`HybridRandNColattice` *(Best Model:* Flexd4f-CVP$_{\gamma_2}$*)*	**158.1**§	**222.1**	**300.7**	**152.3**	**215.8**	**294.5**
Case 2: $B = B_{\mathbf{z}}$						
Value of B	130994	524092	524168	130994	524092	524168
Restricted SVP	**171.5**	**227.7**	**307.9**	-	-	-
`HybridRandNColattice` *(Best Model:* Flexd4f-CVP$_{\gamma_2}$*)*	**169.6**	**226.8**	**308.2**	**163.9**	**220.6**	**302.1**

§: $T_{\texttt{ProBKZ}} = 157.6\log_2(\text{gate})$, $T_{\texttt{RNC}} = 155.8\log_2(\text{gate})$. Call the progressive BKZ-438 reduction and a `HybridRandNColattice` tour with $k = 5$, $(\beta_0, f_0) = (442, 5)$, $\beta_4 = 280$, $(\beta_i, f_i) = (442, 20)$ for $1 \le i \le 3$ and $-\log_2(p_\infty) = 39.3$.

If we consider $\mathbf{B}_\mu = \begin{pmatrix} \mu q\mathbf{I} & \mu\mathbf{A} \\ \mathbf{O} & \mathbf{I} \end{pmatrix}$ as [WXG25], then $B = B_{\mathbf{z}}$ and $\text{Vol} = (\mu q)^n$ with $\mu = B_{\mathbf{z}}/B_{\mathbf{u}}$. We give the security estimation in both cases in Table 3.

Security Estimation for HuFu. The signature forgery attack on HuFu can be formulated as solving an ISIS problem. In the signature scheme, let $(\mathbf{A} = [\mathbf{I}, \hat{\mathbf{A}}, \mathbf{B}], \mathbf{u}) \in \mathbb{Z}_Q^{m\times(2m+n)} \times \mathbb{Z}_Q^m$, where $\hat{\mathbf{A}} \in \mathbb{Z}_Q^{m\times n}$ and $\mathbf{B} \in \mathbb{Z}_Q^{m\times m}$. The signature $\mathbf{x} = (\mathbf{x}_0 + \mathbf{e}, \mathbf{x}_1, \mathbf{x}_2)$ satisfies the equation

$$\mathbf{x}_0 + \mathbf{e} + \hat{\mathbf{A}}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_2 = \mathbf{u} \pmod{Q}.$$

The signature verification process involves computing $\mathbf{x}_0 + \mathbf{e} = \mathbf{u} - \hat{\mathbf{A}}\mathbf{x}_1 - \mathbf{B}\mathbf{x}_2 \pmod{Q}$ and checking $\|(\mathbf{x}_0 + \mathbf{e}, \mathbf{x}_1, \mathbf{x}_2)\| \le B$.

For forgery attack, we should find another $\mathbf{x}$ to pass the verification process. This problem can be regarded as an $\text{ISIS}_{m,m+n,B}$ problem. HuFu paper [YJL+23] reduced it to an approximate CVP and estimate its hardness by

the Nearest-Colattice algorithm [EK20]. The algorithm employs BKZ-β and a Nearest-Colattice with blocksize β to solve the problem. It considers the attack by finding the optimal (β, k) that can minimize the right side of the inequality:

$$B \geq \min_{k \leq m+n} (\delta_\beta^{2m+n-k} Q^{\frac{m}{2m+n-k}}).$$

Here, $\delta_\beta^{2m+n-k} Q^{\frac{m}{2m+n-k}}$ represents the estimated approximate closest vector in a $(2m+n-k)$-dimensional lattice.

However, the estimation in HuFu paper only considers the time cost of the attack as one SVP oracle with block size β, which is a conservative estimation. From Table 2 presented in HuFu paper, we re-estimate its hardness of forgery in Table 4.

Table 4: Security Estimation for HuFu.

Level	Metric	`classical_naive` no `d4f`§	`classical_naive` Consider `d4f`	`classical_matzov22` Consider `d4f`
Lv1	Required Level	**128**	**128**	**143**
	`RandNColattice`	**136.9**	**129.3**	**151.8**
	(Best Model)	D4f-CVP	D4f-CVP	D4f-CVP
Lv3	Required Level	**192**	**192**	**207**
	`RandNColattice`	**208.3**	**198.2**	**224.8**
	(Best Model)	D4f-CVP	D4f-CVP	Prob-BDD
Lv5	Required Level	**256**	**256**	**272**
	`RandNColattice`	**264.8**	**252.9**	**281.4**
	(Best Model)	D4f-CVP	D4f-CVP	Prob-BDD

§: "Consider `d4f`" (resp. "no `d4f`") means BKZ preprocess (resp. not) considering `d4f`.

Security Estimation for one-more-ISIS Blind Signature. The one-more-ISIS problem serves as the foundation for the security of the blind signature scheme [AKSY22]. The blind signature is constructed by the following steps: (1) The user samples $\mathbf{h}' \in \mathcal{R}_{512}$ and short vectors $(\mathbf{x}_1, \mathbf{x}_2)$ with $\|(\mathbf{x}_1, \mathbf{x}_2)\|_\infty \leq 2$, then computes $\mathbf{t} = \mathbf{h}'\mathbf{x}_1 + \mathbf{x}_2 + H(\mu) \mod q_F$ where $H(\mu)$ is the hash of the message μ. (2) The value $\mathbf{t}$ is sent to the signer, who responds by generating a short preimage $\mathbf{y} = (\mathbf{y}_1 \| \mathbf{y}_2)$ through the SamplePre routine such that $\mathbf{h} \cdot \mathbf{y}_1 + \mathbf{y}_2 = \mathbf{t} \mod q$, with $\|\mathbf{y}\|$ bounded by B. $\mathbf{y} = (\mathbf{y}_1 \| \mathbf{y}_2)$ constitutes the signature in this blind signature scheme.

The adversary can forge a valid signature by generating $\mathbf{h}'$ by $\mathbf{h}' = \mathbf{x}_1' \cdot \mathbf{h} + \mathbf{x}_2' \mod q$, where $\|\mathbf{x}_1'\|_\infty \leq 2$ and $\|\mathbf{x}_2'\|_\infty \leq 2$. This enables the construction of an alternative solution $\mathbf{z} = (\mathbf{y}_1 - \mathbf{x}_1\mathbf{x}_1' \| \mathbf{y}_2 - \mathbf{x}_1\mathbf{x}_2' - \mathbf{x}_2)$ that satisfies $(\mathbf{h} \| 1) \cdot \mathbf{z} = H(\mu)$

mod q. The forgery succeeds if the norm of $\mathbf{z}$ satisfies $\|\mathbf{z}\| \leq B$.[7] If transform into the one-more-ISIS problem, the parameters set as follows: $n = m = 512$, $q = 7213$, $B = 4469.52$. It infers that $\gamma = 6.8$. The required security is 128 bits. The security estimation result is as Table 5.

Table 5: Security Estimation for one-more-ISIS Blind Signature.

Level	Metric	`classical_naive`	`classical_matzov22`	
		no `d4f`	Consider `d4f`	
	Required Level	**128**	**128**	**143**
Lv1	`RandNColattice`	**126.6**	**120.5**	**152.9**
	(Best Model)	D4f-CVP	D4f-CVP	Prob-BDD

7 Experiments for Correctness Verification

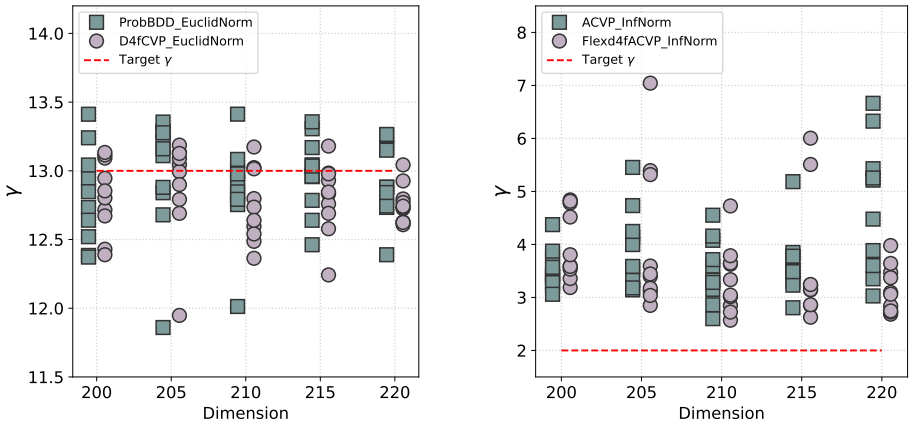
Since [KKNM25] implements the most efficient sieving-based `IterSlicer` and $\texttt{RandSlicer}_{\gamma_1}$ based on G6K in C++/Cython/Python[8], we implemented our `RandNColattice`[9] algorithm in C++/Cython/Python based on their work. The purpose of this experiment is to validate the distance models developed in Secs. 5.2 and 5.3. In particular, the experiment evaluates the accuracy of the predicted output distance of `RandNColattice`, rather than the computational cost of the attack. The experiments were conducted on two hardware platforms: a server equipped with an Intel Xeon Gold 5218 CPU @ 2.30GHz (64 threads) and dual NVIDIA GeForce RTX 3090 GPUs, as well as a desktop computer featuring an Intel Core i5-9500 CPU @ 3.00GHz. To ensure consistency and isolate algorithmic behavior, all tests were executed using a single thread.

In Fig. 2, we evaluate the correctness of the estimated distance bounds for each strategy model in `RandNColattice`among lattices with dimension $d \in \{200, 205, 210, 215, 220\}$. Let D_{out} denote the output distance, and define the approximation factor shown in the figure as $\gamma = \frac{D_{\text{out}}}{\mathsf{GH}(\mathcal{L})}$. Note that D_{out} varies depending on the chosen norm. For the Euclidean norm (Fig. (2a)), we generated strategies aiming for a target of $\gamma = 12$, and the empirical outputs closely match the required approximation factor. Conversely, for the infinity norm case (Fig. (2b)), the strategies generated to target $\gamma = 2$ fail to meet the expected bound. We attribute this discrepancy to two primary reasons: (1) the Gaussian probability estimates applied to the infinity norm are overly optimistic; (2) the

[7] The original scheme analyzed a more relaxed condition $\|\mathbf{z}\| \leq \sqrt{B^2 + 24 \cdot 512 \cdot 4 + 22 \cdot 512 \cdot 2}$, which cannot make the forgery succeed.

[8] https://github.com/ElenaKirshanova/g6k_hybrid/tree/ac_artifact

[9] https://anonymous.4open.science/r/hybrid_randomized_nearest_colattice-27F4



(a) ℓ_2 norm. (b) ℓ_∞ norm.

Fig. 2: Distance Estimation for different strategy models in `HybridRandNColattice`. (1) "ProbBDD_EuclidNorm": the Prob-BDD model; (2) "D4fCVP_EuclidNorm": the D4f-CVP model; (3) "ACVP_InfNorm": the CVP_{γ_2} model; (4) "Flexd4f_InfNorm": the $\mathsf{Flexd4f\text{-}CVP}_{\gamma_2}$ model.

current model neglects the actual geometric properties of the output, failing to account for the fact that the underlying distribution inherently forms an ellipsoid rather than a regular hypercube.

8 Open Problem

Compute and increase the success probability under ℓ_∞-norm. We currently estimate the exact success probability $p_\infty = \mathrm{Vol}(\mathcal{L} \cap V \cap [-D, D]^d)/\mathrm{Vol}(\mathcal{L} \cap V)$ with V satisfying Eq. (5.2), by a simplified model. A large open problem, in several works related to the ∞-norm, is to compute or bound the exact volume intersection, either by theoretical or by numerical methods. Since computing p_∞ is intrinsically equivalent to evaluating the volume of the intersection $\mathrm{Vol}(V \cap [-D, D]^d)$, a second interesting problem is how we can maximize this volume. For example, one can design a specific block structure where earlier blocks employ smaller dimensions and later blocks use larger ones. This configuration is possibly geometrically advantageous because the initial low-dimensional blocks with their highly skewed orientation are more likely to be fully contained within the hypercube $[-D, D]^d$ (as a β-dimensional subspace can accommodate a ball of radius up to $\sqrt{\frac{d(d+1)}{\beta(\beta+1)}} \cdot D$ [Bal92]). Consequently, the subsequent higher-dimensional blocks, which correspond to shorter radii, could remain within the ℓ_∞ bound with a significantly higher probability.

Acknowledgement. This work is supported by the National Cryptologic Science Fund of China under Grant No. 2025NCSF01005. Wenwen Xia is supported by

France 2030 under the French National Research Agency award number "ANR-22-PNCQ-0002" (HQI initiative).

References

AGPS20. Martin R Albrecht, Vlad Gheorghiu, Eamonn W Postlethwaite, and John M Schanck. Estimating quantum speedups for lattice sieves. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*, pages 583–613. Springer, 2020.

AKSY22. Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 39–53, 2022.

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, October 2015.

Bab86. L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, March 1986.

Bal92. Keith Ball. Ellipsoids of maximal volume in convex bodies. *Geometriae Dedicata*, 41(2):241–250, 1992.

BDGL16. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 10–24. SIAM, 2016.

BDK$^+$21. Bai, Shi, Léo Ducas, Eike Kiltz, Lepoint, Tancrède, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Dilithium, 2021.

BN24. Henry Bambury and Phong Q Nguyen. Improved provable reduction of ntru and hypercubic lattices. In *International Conference on Post-Quantum Cryptography*, pages 343–370. Springer, 2024.

CN11. Yuanmi Chen and Phong Q Nguyen. Bkz 2.0: Better lattice security estimates. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–20. Springer, 2011.

DEP23. Léo Ducas, Thomas Espitau, and Eamonn W Postlethwaite. Finding short integer solutions when the modulus is small. In *Annual International Cryptology Conference*, pages 150–176. Springer, 2023.

DLdW19. Emmanouil Doulgerakis, Thijs Laarhoven, and Benne de Weger. Finding closest lattice vectors using approximate voronoi cells. In *Post-Quantum Cryptography: 10th International Conference, PQCrypto 2019, Chongqing, China, May 8–10, 2019 Revised Selected Papers 10*, pages 3–22. Springer, 2019.

DLvW20. Léo Ducas, Thijs Laarhoven, and Wessel PJ van Woerden. The randomized slicer for cvpp: sharper, faster, smaller, batchier. In *IACR International Conference on Public-Key Cryptography*, pages 3–36. Springer, 2020.

dPT26. Paola de Perthuis and Filip Trenkić. Refined modelling of the primal attack, and variants against module-lwe. In Magali Bardet and Ruben Niederhagen, editors, *Post-Quantum Cryptography*, pages 342–376, Cham, 2026. Springer Nature Switzerland.

Duc18. Léo Ducas. Shortest vector from lattice sieving: a few dimensions for free. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 125–145. Springer, 2018.

EK20. Thomas Espitau and Paul Kirchner. The nearest-colattice algorithm: Time-approximation tradeoff for approx-cvp. *Open Book Series*, 4(1):251–266, 2020.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008.

KKNM25. Alexander Karenin, Elena Kirshanova, Julian Nowakowski, and Alexander May. Fast slicer for batch-cvp: Making lattice hybrid attacks practical. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 100–132. Springer, 2025.

Kle00. Philip Klein. Finding the closest lattice vector when it's unusually close. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, page 937–941, USA, 2000. Society for Industrial and Applied Mathematics.

Laa20. Thijs Laarhoven. Approximate voronoi cells for lattices, revisited. *Journal of Mathematical Cryptology*, 15(1):60–71, 2020.

MAT22. MATZOV. Report on the Security of LWE: Improved Dual Lattice Attack. https://doi.org/10.5281/zenodo.6412487, April 2022.

MR07. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM journal on computing*, 37(1):267–302, 2007.

Nat23. National Institute of Standards and Technology (NIST). Security evaluation criteria. NIST Post-Quantum Cryptography Standardization Project, 2023. Accessed: 2023-11-01.

Ngu09. Phong Q Nguyen. Hermite's constant and lattice algorithms. In *The LLL Algorithm: Survey and Applications*, pages 19–69. Springer, 2009.

PMHS19. Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. Approx-svp in ideal lattices with pre-processing. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 685–716. Springer, 2019.

Sch03. Claus Peter Schnorr. Lattice reduction by random sampling and birthday methods. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 145–156. Springer, 2003.

SFS09. Naftali Sommer, Meir Feder, and Ofir Shalvi. Finding the closest lattice point by iterative slicing. *SIAM Journal on Discrete Mathematics*, 23(2):715–731, 2009.

vW23. Wessel van Woerden. *Lattice cryptography: from cryptanalysis to New Foundations*. PhD thesis, Leiden University, 2023.

WXG25. Geng Wang, Wenwen Xia, and Dawu Gu. Heuristic algorithm for solving restricted svp and its applications. In *International Conference on Post-Quantum Cryptography*, pages 119–152. Springer, 2025.

YJL+23. Yang Yu, Huiwen Jia, Leibo Li, Delong Ran, Zhiyuan Qiu, Shiduo Zhang, Xiuhan Lin, and Xiaoyun Wang. Hufu: Hash-and-sign signatures from powerful gadgets. 2023.

A Cost Model

In our estimation framework, we instantiate the SVP oracle for BKZ preprocessing with the BDGL sieve algorithm [BDGL16], and the CVP_{γ_1} oracle ($\gamma_1 \geq 1$) with the $\texttt{RandSlicer}_{\gamma_1}$ approach [DLvW20] (Alg. 2). We chose these specific algorithms because they currently represent the most efficient solvers for the SVP and CVP_{γ_1} problems, respectively.

We categorize the estimation into two types: theoretical estimation (denoted as `naive_classical` cost model) and gate count estimation based on the classical circuit models AGPS20 [AGPS20] and MATZOV22 [MAT22].

Theoretical Estimation. From Eq. (4) in [DLvW20], the theoretical cost of a single call to the iterative slicer is given by:

$$T_{\texttt{IterSlicer}}(\alpha, d) = \left(\alpha \cdot \sqrt{1 - \frac{2(1-1/\alpha^2)}{1+\sqrt{1-1/\alpha^2}}}\right)^{-d+o(d)}, \tag{18}$$

where for the theoretical model the $o(d)$ term is subsequently ignored. This cost increases monotonically with the parameter α. Meanwhile, the algorithm returns an approximate closest vector within a factor of $\gamma_1 = \frac{\alpha^2}{2\sqrt{\alpha^2-1}}$ [Laa20], which decreases as α increases. The choice of $\alpha = \sqrt{4/3}$ represents a critical trade-off: while a larger α reduces the time per iterative slicer call $T_{\texttt{IterSlicer}}$ (due to a lower requirement on the number of short vectors), it demands that the sieve algorithm enumerate lattice vectors up to the $(\frac{4}{3})^{d/2}$-th shortest one to guarantee vector quality. Therefore, $\alpha = \sqrt{4/3}$ is chosen to balance the computational complexity of the `IterSlicer` tour against the approximation factor γ_1. For this value, the time complexity simplifies to:

$$T_{\texttt{IterSlicer}}(\sqrt{4/3}, d) = \left(\frac{9}{8}\right)^{d/2} \approx 2^{0.08496d}. \tag{19}$$

and $\gamma_1 = \sqrt{4/3}$. In the following description, we assume $\alpha = \gamma_1 = \sqrt{4/3}$ as a default setting.

To formalize our theoretical complexity analysis, we establish the following framework and parameters:

- **SVP Preprocessing Cost**: Following the best known asymptotic complexity for lattice sieving algorithms [BDGL16], the time cost of a single SVP call is given by:
$$T_{\texttt{Sieve}}(d) = (3/2)^{d/2} \approx 2^{0.292d}. \tag{20}$$
- $\texttt{RandSlicer}_{\gamma_1}$ **Cost**: As introduced in Sec. 2.5, the time cost of an $\texttt{RandSlicer}_{\gamma_1}$ tour should be
$$T_{\texttt{RS}}(d, M, \gamma_1) = \frac{M}{\mathbb{P}_{\alpha^2, \gamma_1^2}(d)} T_{\texttt{IterSlicer}}(d), \tag{21}$$
where M is the batch size of the target vectors.

Note that in [DLvW20], in order to reach the time complexity for a single target as stated, the memory usage used for preprocessing the nearest-neighbor could be required to be significantly higher than α^d. However, when $M/\mathbb{P}_{\alpha^2,\gamma_1^2}(d) \geq \alpha^d$, [DLvW20] explains that one can reach the claimed complexity with only α^d memory by processing them as a batch. We can assume without loss of generality that we are in this regime, as otherwise the preprocessing cost $T_{\text{sieve}}(d)$ would dominate regardless.

Gate Count Estimation. For SVP gate count estimation, we use the classical circuit models AGPS20 [AGPS20] and MATZOV22 [MAT22] in the lattice estimator [APS15], where AGPS20 is proposed in [AGPS20] and MATZOV22 is an improvement over [AGPS20].

We analyze the CVP gate count estimation following [AGPS20], which treats the `popcount` operation as a fundamental component. In the `IterSlicer` (Alg. 1, line 4), each query requires a nearest neighbor search (NNS), and the total time complexity equals the product of a single NNS cost and the optimal number of arbitrary-step paths per `IterSlicer` tour. The concrete analysis is as follows:

Nearest Neighbor Search. The `popcount` operation serves as a fundamental component in Nearest Neighbor Search (NNS) algorithms. Following the implementation proposed by [AGPS20], it provides the computational costs of `popcount` for both classical and quantum computers.

- Let $T_{\texttt{pop}}$ denote the time cost for one `popcount` operation. We adopt both classical and quantum gate counts for $T_{\texttt{pop}}$ as proposed in [AGPS20].
- Let $T_{\texttt{dot}}$ represent the inner product computation cost for vectors passing the `popcount` filter.
- Let p be the probability that a vector pair passes the `popcount` filter.
- Let L be the preprocessed sieving set generated from lattice basis $\mathbf{B}$ and $N = |L|$ denote the size of the sieving set.

The total computational cost for one NNS query to find the nearest neighbor of target vector $\mathbf{t}$ is given by $T_{\text{NNS}} = (p \cdot T_{\texttt{dot}} + T_{\texttt{pop}}) \cdot N$. In the `popcount` implementation described in [AGPS20], the $T_{\texttt{dot}}$ term is omitted from the complexity analysis. This simplification yields the final NNS process complexity:

$$T_{\text{NNS}} = T_{\texttt{pop}} \cdot N. \tag{22}$$

In the `IterSlicer` algorithm, NNS is called to identify the approximate vector $\mathbf{w} \in L$ to a target vector $\mathbf{t}$. We estimate the cost of the `IterSlicer` by Heuristic 7, which is based on Heuristic Claim 3 in [DLvW20] and we add the cost of sieving preprocess for one `IterSlicer` time cost. The size of the sieve set determines both the time cost of sieving and the approximation ratio of the output from the `IterSlicer` tour.

Heuristic 7 (A modification of Claim 3 in [DLvW20]) *Given a list L of the $N = \alpha^d \leq (4/3)^{d/2}$ shortest lattice vectors as input, we can solve a single CVP_{γ_1} instance with $\gamma_1 = \frac{\alpha^2}{2\sqrt{\alpha^2-1}}$ and the following complexities:*

$$T_{\mathit{IterSlicer}}(\alpha, d) = N \cdot t \cdot T_{\mathrm{pop}} = \left(\alpha \cdot \sqrt{1 - \frac{2 \cdot (1 - 1/\alpha^2)}{1 + \sqrt{1 - 1/\alpha^2}}}\right)^{-d} \cdot T_{\mathrm{pop}}, \tag{23}$$

where t is the optimal number of arbitrary-step paths for each `IterSlicer` *tour.*