Heuristic Algorithm for Solving Restricted SVP and its Applications

Geng Wang[1,2], Wenwen Xia[1,2], and Dawu Gu[1(✉)]

[1] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
{wanggxx, xww_summer, dwgu}@sjtu.edu.cn
[2] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China

Abstract. In lattice-based cryptography, many attacks are performed by finding a short enough vector on a specific lattice. However, it is possible that length is not the only restriction on the vector to be found. A typical example is SVP with infinity norm: since most SVP solving algorithms only aim to find short vector under Euclidean norm, the infinity norm is in fact another restriction on the vector. In the literature, such problems are usually solved by performing exhaustive search on a list of short vectors generated from lattice sieving. However, the sieving list might either be too large or too small to pass the additional restriction, which makes the solving algorithm inefficient in some cases.
Our contribution in this work is as follows: (1) We formally define a new lattice hard problem called restricted SVP, and show that it can be used to generalize many lattice hard problems, including SVP with non-Euclidean norm and Kannan's embedding on approximate CVP. (2) We extend the dimension for free technique and the enumerate-then-slice technique into approximate SVP where the goal is to output a list of short vectors with a certain size. (3) We give the heuristic algorithm for solving restricted SVP, and design a hardness estimator for this algorithm, which can be used to estimate the concrete hardness of signature forgery in Dilithium and other lattice-based signatures. Using this estimator, we present a concrete security analysis for Dilithium against signature forgery under the gate-count model for the first time. Our estimation matches well with the security estimation from core-SVP model in the document of Dilithium, and we also combine our estimator with the rescaling technique to generate a tighter estimation.

Keywords: Lattice-based Cryptography · Shortest Vector Problem · Dilithium · Post-quantum cryptography

1 Introduction

Due to its ability to resist quantum attacks and other nice properties such as homomorphism, lattice-based cryptography has received a lot of attention these years. In the post-quantum standardization proposed by NIST, two lattice-based algorithms Kyber [11] and Dilithium [13] have been chosen as recommended

standards. Since the security of these algorithms relies on the hardness of certain lattice problems, it becomes important to design more efficient algorithms for solving these problems.

There are two basic types of lattice problems, namely the shortest vector problem SVP and the closest vector problem CVP, both of them have many different variants. For the SVP family, we have exact SVP, which aims to find the shortest non-zero vector in a lattice, approximate SVP, which aims to find a short enough (but not necessarily shortest) vector in a lattice, and unique SVP, which aims to find the shortest non-zero vector in a lattice which satisfies some specific conditions. Many lattice problems used in constructing post-quantum cryptosystems, such as SIS [2] or LWE [37], can be solved through one of these problems.

All these problems consider only the length of vectors, usually in the Euclidean norm. However, there are still some related problems not only aim to find a short lattice vector, but also require the vector to satisfy some additional properties, which relate but not totally depend on the length of the vector. Those existing definitions cannot properly describe such type of problems.

In this paper, we present a new lattice hard problem, called restricted SVP (Definition 4.2), which can be viewed as a more flexible version of approximate SVP. Informally, restricted SVP aims to find a lattice vector $\mathbf{v}$ which satisfies a restriction $\mathcal{R}$ which is a predicate, such that $\Pr(\mathcal{R}(\mathbf{v}) = 1 : \mathbf{v} \in \mathcal{L} \wedge \|\mathbf{v}\| \leq len) = 1$ increases as len decreases.

Restricted SVP can be used to generalize a large type of lattice problems. We list two of them:

(1) Kannan's embedding on approximate CVP.

Kannan's embedding technique [29] can be used to solve CVP-type lattice problems as follows: for a d-dimensional lattice basis $\mathbf{B}$ and a target vector $\mathbf{t}$, Kannan's embedding aims to find a short vector on the lattice $\mathcal{L}' = \mathcal{L}\begin{pmatrix} \mathbf{B} & \mathbf{t} \\ \mathbf{0}^T & M \end{pmatrix}$, M is called the embedding factor, and if the short vector has the form $\begin{pmatrix} \mathbf{x} \\ \pm M \end{pmatrix}$, then $\mathbf{x} = \pm(\mathbf{t} - \mathbf{v})$ where $\mathbf{v}$ is a close lattice vector to $\mathbf{t}$.

Kannan's embedding has proven to be useful in solving gapCVP and related problems, such as BDD or LWE [6]. However, Kannan's embedding offers only a non-tight reduction from approximate CVP to approximate SVP. The main reason is that, for gapCVP, the shortest vector in $\mathcal{L}'$ must have $\pm M$ as its last element, but for approximate CVP, we only find an approximate shortest vector for $\mathcal{L}'$, which last element could be $c \cdot M$ for any $|c|$ small enough. While a short non-zero vector in $\mathcal{L}'$ may have its last element either 0 or $c \cdot M$ for $|c| \geq 2$, a correct solution for CVP requires the last element to be $\pm M$, which cannot be promised for approximate SVP.

Under our new definition, solving approximate CVP by Kannan's embedding can be turned into solving restricted SVP with restriction: $v_d = \pm M$.

(2) Approximate SVP under infinity norm.

Many lattice-based algorithms, including the NIST standard Dilithium [13] have their hardness based on lattice problems with infinity norm. However, the

most efficient lattice solving algorithms only work on lattice problems with Euclidean norm. Although there are some researchers which discussed on lattice problems on other norms [1, 18, 34], none of their algorithms are as efficient as the algorithms on Euclidean norm.

In this paper, we consider the problem of solving approximate SVP under infinity norm with bound B, which can be viewed as solving restricted SVP with restriction: $\|\mathbf{v}\|_\infty \leq B$. Also, by combining the restriction with the restriction on Kannan's embedding for approximate CVP, we can see that approximate CVP under infinity norm can also be turned into an instance for restricted SVP. Since signature forgery for Dilithium and many other lattice signatures is in fact a problem for solving approximate CVP under infinity norm, we can use a solving algorithm for restricted SVP to perform forgery attack on lattice signatures or estimate their security levels.

Lattice Solving Algorithms for Approximate SVP.

Basic algorithms for solving SVP (sometimes called SVP oracle) include enumeration [3, 4, 29] and lattice sieving [16, 17, 36], which are highly efficient for solving exact SVP or approximate SVP with small factors (e.g. $\gamma = 1.05$). These algorithms have their time costs exponential in the lattice dimension d. The most efficient sieving algorithm BDGL [16] has a theoretical complexity of $\sqrt{3/2}^{d+o(d)} \approx 2^{0.292(d+o(d))}$, which has proven to be the theoretical lower bound for sieving type algorithms [30]. However, the problems used in lattice-based cryptography usually have larger approximation factors (also with larger dimensions), which are inefficient to solve using solely SVP oracles.

More commonly used algorithms in cryptanalysis of lattice-based cryptography are lattice reduction algorithms, including LLL [32] and BKZ [38] (or their variants). While LLL algorithm only solves approximate SVP with exponentially large approximation factors, BKZ is more flexible since it has a tunable parameter called blocksize β, where larger β can solve approximate SVP with smaller approximation factor, but with longer time. BKZ calls the SVP oracle with dimension β, hence the time cost of BKZ is exponential in β. There are many works on improvements for BKZ, including [9, 21, 39], which further improved its efficiency.

Also in solving lattice challenges (see www.latticechallenge.org), Sun et al. used a combination method of BKZ and sieving, which we refer as a two-step mode in this paper (we shall specify it later in Section 3.2).

It is not hard to see that any approximate SVP solving algorithm can be viewed as a probabilistic algorithm for solving restricted SVP: heuristically assume that an algorithm can find a uniform lattice vector of length $\leq len$, then the algorithm can be used to solve the restricted SVP problem with probability $\Pr(\mathcal{R}(\mathbf{v}) = 1 : \mathbf{v} \in \mathcal{L} \wedge \|\mathbf{v}\| \leq len)$.

At a first glimpse, since we consider general restrictions, the only meaningful method for finding a lattice vector passing the restriction is through exhaustive search, so the optimal strategy for solving such a problem should be simply repeating the (probabilistic) approximate SVP solving algorithm many times. However, we show that it is in fact not the case: by a careful design on the

algorithm, we can reuse most of the intermediate results in an approximate SVP solving algorithm, and perform the exhaustive search only on the last step. So by designing an optimized algorithm specifically for restricted SVP, we can solve the problem faster and get a tighter bound for lattice algorithms which hardness based on restricted SVP.

1.1 Our Contribution

The main contributions of this work is as follows:

(1) We formally define a new type of lattice hard problem called restricted SVP in Section 4, which can be used to generalize a large fraction of lattice-related problems, including Kannan's embedding on approximate CVP or approximate SVP with infinity norm. We also show that restricted SVP can be solved by a two-step lattice solving algorithm for approximate SVP which outputs a set of lattice short vectors of a specific size determined by the restriction in the problem.

(2) We design two variants of two-step algorithms for solving approximate SVP in Section 5, namely flexible dimension for free (Section 5.1) and sieve-then-slice (Section 5.2). The first algorithm is used to output a list of short vectors which size is smaller than $\sqrt{4/3}^{\kappa}$, where κ is the sieving dimension, and the second algorithm is used to output a list of short vectors which size is larger than $\sqrt{4/3}^{\kappa}$. Then we combine the two algorithms to get a solving algorithm for restricted SVP in Section 5.3, and discuss its time complexity.

(3) We give approximations for the probabilities that a lattice vector could pass the restrictions in the two applications of restricted SVP: Kannan's embedding on approximate CVP in Section 6.1 and approximate SVP with infinity norm in Section 6.2. By combining these two results, we can give an estimation for hardness of approximate CVP with infinity norm and its related problems, such as the MISIS problem, which is a CVP-type hard problem on module lattices [31] used in Dilithium. We use this new hardness estimator to estimate the security for unforgeability in Dilithium in Section 6.3, and present its security bits under the gate-count model, which is absent in the current document of Dilithium.

2 Preliminaries

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 $i+1$-th column vector. The Euclidean norm of a vector $\mathbf{v} = (v_0, ..., v_{m-1})^T \in \mathbb{R}^m$ is denoted by $\|\mathbf{v}\| = \sqrt{\sum_{i=0}^{m-1} v_i^2}$, and the infinity norm of $\mathbf{v}$ is denoted by $\|\mathbf{v}\|_\infty = \max |v_i|$. We define $[a, b] = \{a, a+1, ..., b-1\}$ and $[d] = [0, d]$.

We use $\mathcal{B}^d(r) \subset \mathbb{R}^d$ to denote a d-dimensional hyperball with radius r, which means that $\mathcal{B}^d(r) = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq r\}$.

Let $\mathcal{N}(\mu, \sigma)$ be the Gaussian distribution with mean value μ and standard deviation σ. $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ stands for the high-dimensional Gaussian with mean value $\boldsymbol{\mu}$ and covariance matrix $\Sigma^T\Sigma$. We also write $\mathcal{U}[a, b]$ as uniform distribution on $[a, b]$.

2.1 Basic Definition on Lattice

Definition 2.1 (Lattice). *Let $\mathcal{B} \in \mathbb{R}^{d\times k}$, $d \geq k$ be a matrix of rank k. The lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ is defined as $\{\mathbf{B}\cdot\mathbf{c} : \mathbf{c} \in \mathbb{Z}^k\}$. If $k = d$, we call $\mathcal{L}$ a full-rank lattice.*

For a full-rank lattice, we define its volume by: $\mathrm{vol}(\mathcal{L}) = |\mathbf{B}|$.

Next, we give the definitions on basic lattice hard problems: SVP and CVP.

Definition 2.2 (Shortest Vector Problem (SVP)). *Given a lattice $\mathcal{L}$, the shortest vector problem finds a vector $\mathbf{x} \in \mathcal{L}$ such that $\|\mathbf{x}\| = \lambda_1(\mathcal{L})$, where $\lambda_1(\mathcal{L}) = \min\{\|\mathbf{v}\| : \mathbf{v} \in \mathcal{L} - \{\mathbf{0}\}\}$.*

Definition 2.3 (Closest Vector Problem (CVP)). *Given a lattice $\mathcal{L}$ and $\mathbf{t} \in \mathbb{R}^d$, the closest vector problem finds a vector $\mathbf{x} \in \mathcal{L}$ such that $\|\mathbf{t}-\mathbf{x}\| \leq \|\mathbf{t}-\mathbf{v}\|$ for any $\mathbf{v} \in \mathcal{L}$.*

We usually consider their approximated version, called approximate SVP (γ-SVP) and approximate CVP (γ-CVP). Their definitions are as follows:

Definition 2.4 (Approximate SVP). *Given a lattice $\mathcal{L}$ and the approximation factor γ, approximate SVP finds a vector $\mathbf{x} \in \mathcal{L}$ such that $\|\mathbf{x}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$.*

Definition 2.5 (Approximate CVP). *Given a lattice $\mathcal{L}$ and $\mathbf{t} \in \mathbb{R}^d$, approximate CVP finds a vector $\mathbf{x} \in \mathcal{L}$ such that $\|\mathbf{t} - \mathbf{x}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$.*

However, determining $\lambda_1(\mathcal{L})$ means that we have already found the shortest vector, which is infeasible in most cases. In lattice solving algorithms, the following Gaussian Heuristic is used to give an approximation of $\lambda_1(\mathcal{L})$:

Definition 2.6 (Gaussian Heuristic).

The Gaussian Heuristic claims that for any convex shape $\mathcal{C} \subset \mathbb{R}^d$ where $\mathrm{vol}(\mathcal{C}) \geq \mathrm{vol}(\mathcal{L})$, there is approximately $\frac{\mathrm{vol}(\mathcal{C})}{\mathrm{vol}(\mathcal{L})}$ lattice points in $\mathcal{C}$.

Specifically, a hyperball with radius $\lambda_1(\mathcal{L})$ contains exactly 1 lattice point, thus the Gaussian Heuristic predicts:

$$\lambda_1(\mathcal{L}) \approx gh(\mathcal{L}) = \sqrt{\frac{d}{2\pi e}}|\mathbf{B}|^{1/d}$$

from the approximation of the volume of a hyberball: $\mathrm{vol}(\mathcal{B}^d(r)) \approx \frac{1}{\sqrt{2\pi}}(\sqrt{\frac{2\pi e}{d}} \cdot r)^d$.

Gaussian Heuristic is the most important heuristic assumption in lattice-based cryptanalysis, which correctness has been verified in some existing works [19].

2.2 Lattice Reduction

Next, we introduce lattice reduction, which is an important type of lattice solving algorithms. We first give the definition of projected sub-lattice.

Definition 2.7 (Projected Sub-Lattice). *Let* $\mathbf{B}$ *be a lattice basis of the full rank lattice* $\mathcal{L}$*. For* $i \in [d]$*, we denote by* π_i *the orthogonal projection over* $(\mathbf{b}_0, ..., \mathbf{b}_{i-1})^{\perp}$*, which means that for any* $\mathbf{v} \in \mathbb{R}^d$*,* $\pi_i(\mathbf{v})^T \mathbf{b}_j = 0$ *for any* $j < i$ *and* $\mathbf{v} - \pi_i(\mathbf{v}) \in \text{span}(\mathbf{b}_0, ..., \mathbf{b}_{i-1})$*.*

For $0 \leq j \leq k \leq d-1$*, we denote by* $\mathbf{B}[j,k]$ *the local projected block* $(\pi_j(\mathbf{b}_j), \pi_j(\mathbf{b}_{j+1}), ..., \pi_j(\mathbf{b}_{k-1}))$*, and by* $\mathcal{L}[j,k]$ *the lattice spanned by* $\mathbf{B}[j,k]$*.* $\mathcal{L}[j,k]$ *is called a projected sub-lattice of* $\mathcal{L}$*.*

Projected sub-lattice can be more easily expressed from the Gram-Schmidt orthogonalized basis $\mathbf{B}^* = (\mathbf{b}_0^*, ..., \mathbf{b}_{d-1}^*)$ as follows:

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=0}^{i-1} \mu_{i,j} \mathbf{b}_j^*,$$

$$\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2, i < j < d.$$

It directly follows that $\mathbf{b}_j^*, ..., \mathbf{b}_{k-1}^*$ is the Gram-Schmidt orthogonalized basis of $\mathcal{L}[j,k]$.

The volume of a lattice $\text{Vol}(\mathcal{L})$ can be calculated from $\mathbf{B}^*$ as:

$$\text{Vol}(\mathcal{L}) = |\mathbf{B}| = \prod_{i=0}^{d-1} \|\mathbf{b}_i^*\|.$$

Definition 2.8 (Size-reduced). *A basis* $\mathbf{B}$ *is size-reduced if the following holds: For* $1 \leq j < i \leq d : |\mu_{i,j}| \leq 1/2$*.*

Definition 2.9 (HKZ and BKZ reductions [35]). *The basis* $\mathbf{B}$ *of a* d*-dimensional lattice* $\mathcal{L}$ *is HKZ reduced if* $\mathbf{B}$ *is size-reduced and* $\|\mathbf{b}_i^*\| = \lambda_1(\mathcal{L}[i:d])$*, for all* $i < d$*. A* d*-dimensional* $\mathcal{L}$ *is BKZ-*β *reduced if* $\mathbf{B}$ *is size-reduced and* $\|\mathbf{b}_i^*\| = \lambda_1(\mathcal{L}[i : \min\{i+\beta, d\}])$*, for all* $i < d$*.*

The BKZ algorithm [21,38] with blocksize β can generate a BKZ-β reduced lattice basis after running a sufficient number of tours. But even the number of tours is not enough, it may still output a BKZ-β' reduced basis for some $\beta' < \beta$. We do not go into details here.

Definition 2.10 (Root Hermite Factor). *For a basis* $\mathbf{B}$ *of* d*-dimensional lattice, the root Hermite factor is defined as*

$$\delta(\mathbf{B}) = \left(\|\mathbf{b}_0\| / |\mathbf{B}|^{1/d} \right)^{1/d}.$$

For larger blocksize of BKZ, root Hermite factor of a BKZ-β reduced basis follows the asymptotic formula [20]:

$$\delta(\beta)^{2(\beta-1)} = \frac{\beta}{2\pi e}(\beta\pi)^{1/\beta}.$$

$\delta(\mathbf{B})$ can be used to measure current lattice basis quality of the lattice basis $\mathbf{B}$. A better lattice basis quality of $\mathbf{B}$ corresponds to a smaller $\delta(\mathbf{B})$.

Definition 2.11 (Geometric Series Assumption [5]). *Let* $\mathbf{B}$ *be a BKZ-*β *reduced basis with sufficiently large* β*, then* $\|\mathbf{b}_i^*\| \approx \alpha \cdot \|\mathbf{b}_{i-1}^*\|$, $0 < \alpha < 1$.

The geometric series assumption (briefly GSA) is another widely used heuristic assumption in lattice-based cryptanalysis. Combining GSA with root-Hermite factor (Definition 2.10) and $\mathrm{Vol}(\mathcal{L}) = \prod_{i=0}^{d-1} \|\mathbf{b}_i^*\|$, it infers that $\alpha = \delta^{-\frac{2d}{d-1}} \approx \delta^{-2}$.

2.3 Basic Lattice Algorithms

Babai's Lifting. Babai's nearest plane algorithm, also called Babai's lifting, can be used to solve approximate CVP either the approximation factor is exponentially large or the target vector is extremely close to a lattice vector. Babai's lifting is a basic component of many lattice solving algorithms. We describe the algorithm as follows:

input : A target vector $\mathbf{t} \in \mathbb{R}^d$, lattice basis $\mathbf{B}$
output: The close vector $\mathbf{w} \in \mathcal{L}$ s.t. $\|\mathbf{t} - \mathbf{w}\| \leq 2^{d/2}\mathrm{dist}(\mathbf{t}, \mathcal{L})$;
Function `Babai-Lift(t, B):`
  $\mathbf{B}^* \leftarrow$ Gram-Schmidt basis of $\mathbf{B}$;
  for j **from** $d-1$ **to** 0 **do**
    $c_j \leftarrow \lceil \frac{\langle \mathbf{t}, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \rfloor$, $\mathbf{t}' \leftarrow \mathbf{t} - c_j \mathbf{b}_j$;
  return $\mathbf{t} - \mathbf{t}'$;

Algorithm 1: Babai's Nearest Plane Algorithm [12]

Babai's lifting is usually used to find a lattice vector $\mathbf{v}$ on the full lattice $\mathcal{L}$ which is close to a certain vector $\mathbf{v}'$ in a projected sub-lattice $\mathcal{L}[j:k]$. This operation is called "lift", which is exactly the algorithm named after.

Lattice Sieving and Dimension for Free. A lattice sieving algorithm proceeds as follows: first sample a relatively large set L of lattice vectors, for each two vectors $\mathbf{u}, \mathbf{v} \in L$, if $\|\mathbf{u} \pm \mathbf{v}\| < \|\mathbf{u}\|$ or $\|\mathbf{v}\|$, use $\mathbf{u} \pm \mathbf{v}$ to replace $\mathbf{u}$ or $\mathbf{v}$, repeat this procedure until no vector pair in L can be reduced. If the size of L satisfies $|L| \geq \sqrt{4/3}^{d+o(d)}$, lattice sieving can output a set of shortest lattice vectors.

There are different types of lattice sieving which find vector pairs in different methods. The most efficient sieving algorithm is BDGL sieve [16] which costs $\sqrt{3/2}^{d+o(d)}$ using the nearest neighbor searching techniques.

Dimension for free (d4f for short) technology [25] can bring sub-exponential time speedup and memory decrease for sieving algorithms. D4f is based on the following fact:

For the shortest non-zero vector $\mathbf{v} \in \mathcal{L}$, $\pi_f(\mathbf{v}) \in \mathcal{L}[f:d]$. Since $\|\pi_f(\mathbf{v})\| \leq \|\mathbf{v}\|$, if the following condition is satisfied:

$$\|\mathbf{v}\| \approx gh(\mathcal{L}) \leq \sqrt{4/3} gh(\mathcal{L}[f:d]),$$

then $\pi_f(\mathbf{v})$ must be inside the sieving output of lattice $\mathcal{L}[f:d]$ which contains $\sqrt{4/3}^{d-f}$ shortest vectors of $\mathcal{L}[f:d]$ (ensured by the Gaussian Heuristic). So it is possible to recover $\mathbf{v}$ by performing Babai's lifting on each vector in the sieving list $L[f:d]$ of the projected sub-lattice $\mathcal{L}[f:d]$. We can solve the equation to find f.

Furthermore, if we heuristically assume that for $\mathbf{v} = \sum_{i=0}^{d-1} v_i^* \mathbf{b}_i^*$, every v_i^* has similar norm, then we can assume that $\|\pi_f(\mathbf{v})\| \approx \sqrt{\frac{d-f}{d}}\|\mathbf{v}\|$ to get a larger number of free dimensions f with the following equation:

$$\sqrt{\frac{d-f}{d}} gh(\mathcal{L}) \leq \sqrt{4/3} gh(\mathcal{L}[f:d])$$

.

The Block Korkine-Zolotarev (BKZ) Algorithm. BKZ is a lattice reduction algorithm introduced by Schnorr and Euchner [38].

The main idea of BKZ algorithm is using β dimensional SVP oracle such as lattice sieving to find a lattice vector $\mathbf{v}_i \in \mathcal{L}[i:i+\beta]$ for each i, s.t. $\|\mathbf{v}_i\| = gh(\mathcal{L}[i:i+\beta])$, lift it into a lattice vector, then insert it in i-th position of lattice basis. After each insertion, it will improve the quality of the lattice basis. Besides, it calls an LLL algorithm before lattice reduction to vanish the linear dependence and make a rough reduction after the insertion. We give a simplified decription for the basic version of BKZ.

input : $\mathbf{B}$, β;
output: $\mathbf{B}$;
Function `BKZ`$(\mathbf{B}, \beta)$**:**
  $\mathbf{B} =$ `LLL`$(\mathbf{B})$;
  for $i \leftarrow 0$ **to** $d-1$ **do**
    $\beta' = \min\{\beta, d-i\}$;
    Find the shortest vector $\mathbf{v}_i^*$ in the projected sub-lattice $\mathcal{L}[i:\beta'+i]$;
    $\mathbf{v}_i \leftarrow$`Babai-Lift`$(\mathbf{v}_i^*)$;
    $\mathbf{B} =$ `LLL`$((\mathbf{b}_0, ..., \mathbf{b}_{i-1}, \mathbf{v}_i, \mathbf{b}_i, ..., \mathbf{b}_{d-1}))$;

Algorithm 2: BKZ

The algorithm above defines a tour of BKZ with blocksize β. To achieve a basis which is BKZ-β reduced, we should run the BKZ-β tour many times. Progressive BKZ can be used to speed up this procedure, where the blocksize β is increased after each tour. In this work, we use the trivial progressive strategy as in [22], where β is increased by 1 after each tour. We note that more flexible strategy can be used to further accelerate the reduction procedure, as shown in [10,39].

3 Some New Lattice Solving Algorithms

In this section, we present two new lattice solving algorithms we shall use throughout this work, which are modification or combination of some existing algorithms.

3.1 Modified Randomized Slicer

Randomized iterative slicer [23,26] is an algorithm for solving CVP with preprocessing (CVPP for short). It relies on enumeration or lattice sieving to generate a list of shortest lattice vectors which is the preprocessing step, and the algorithm itself uses the list to solve CVP for a target vector $\mathbf{t}$. Randomized slicer is the most efficient algorithm in solving CVPP, which costs approximately $\sqrt{\frac{18}{13}}^{d+o(d)}$ for a single CVP instance.

Here we do not introduce the original randomized slicer algorithm in [23], but a modified version which will be used later in our paper.

input : The lattice $\mathcal{L}$, a list of target vectors $T \subset \mathbb{R}^d$, a list of short lattice vectors $L_{in} \subset \mathcal{L}$, size of the output list S;
output: A list of short vectors $L_{out} \subset \mathcal{L} \cup (T + \mathcal{L})$;

Function `RandomizedSlicer`$(\mathcal{L}, T, L_{in}, S)$**:**
  $L_{out} \leftarrow L_{in}$;
  repeat
    $\mathbf{t}' \leftarrow$ `Sample`$(T + \mathcal{L})$;
    for each $\mathbf{r} \in L_{in}$ **do**
      if $\|\mathbf{t}' - \mathbf{r}\| < \|\mathbf{t}'\|$ **then**
        Replace $\mathbf{t}' \leftarrow \mathbf{t}' - \mathbf{r}$ and restart the for-loop;
    $L_{out} \leftarrow L_{out} \cup \{\mathbf{t}'\}$;
  until $|L_{out}| = S$;
  return L_{out};

Algorithm 3: Modified Randomized Slicer

The algorithm is modified from the two aspects: (1) instead of returning lattice vectors, we return a list of short vectors in $\mathcal{L} \cup (T + \mathcal{L})$; (2) the vector $\mathbf{t}'$

is added into the list after each iteration, instead of only returning one shortest $\mathbf{t}'$ for each $\mathbf{t} \in T$ (where $\mathbf{t} - \mathbf{t}'$ is the closest lattice vector of $\mathbf{t}$). We can see that none of these modifications has effect on the time complexity of randomized slicer.

For a lattice with dimension d, let L_{in} be the sieving list which contains $\sqrt{4/3}^d$ short lattice vectors, which lengths heuristically $\leq \sqrt{4/3} \cdot \lambda_1(\mathcal{L})$. By the discussion in [23], we can also heuristically assume that all vectors in L_{out} have lengths $\leq \sqrt{4/3} \cdot \lambda_1(\mathcal{L})$.

3.2 Two-step Algorithm for Solving Approx-SVP

Instead of solving lattice problems by using only lattice reduction algorithms such as BKZ, a more efficient approach is combining lattice reduction with a SVP solver: first reduce the lattice basis, then find the target vector by a single call to the SVP solver (either enumeration or lattice sieving), which is sometimes called a two-step algorithm in the literature.

Although two-step algorithms are usually considered as folklore approaches for solving lattice problems, there are only few works for theoretically analyzing such algorithms. A recent work by Xia et al. [40] presented a two-step estimator for LWE (or uSVP) hardness, but similar analysis for approximate SVP is still missing.

Since our restricted SVP is in fact a variant of approximate SVP, we first give a formal description for two-step algorithm when solving approximate SVP before we go into any further details.

input : Lattice $\mathcal{L}$ with basis $\mathbf{B}$, target RHF δ, sieving dimension κ;
output: A list of short lattice vectors S;
Function `TwoStepSolver`$(\mathcal{L}, \delta, \kappa)$**:**
  Perform BKZ reduction on $\mathcal{L}$ until its basis has RHF δ;
  $S \leftarrow$ `Sieve`$(\mathcal{L}[0:\kappa])$;
  return S;

Algorithm 4: Two-step Solver

Note that in this algorithm, we take the root Hermite factor δ as its input parameter instead of the blocksize β, since we do not specify which type of BKZ algorithm to be used. Other than BKZ with fixed blocksize β, which will generate a lattice with RHF $\delta(\beta)$, we can also use progressive BKZ as in [10] or the pump-and-jump BKZ in G6K [5], and in the latter cases, we do not necessarily have $\delta = \delta(\beta)$. However, given the BKZ parameters, the RHF δ after BKZ reduction can be calculated from a BKZ simulator as in [21,39].

Unlike what has been proven for solving uSVP or LWE [40], two-step mode does not always lead to an increase in efficiency compared with using BKZ only. However, a nice thing about the two-step solver, is that it always returns a set of short lattice vectors instead of only a lattice basis. This property is useful in our further discussion of restricted SVP solver.

We can see that the two core parameters for a two-step algorithm is the basis quality after lattice reduction, as well as the dimension of the final call to SVP

oracle. So in order to optimize the algorithm for an approximation factor γ, we must find (β, κ) such that $gh(\mathcal{L}[0:\kappa])/gh(\mathcal{L}) \leq \gamma$ and the total time cost for BKZ-β and sieving with dimension κ is minimal (we suppose that the lattice basis has RHF δ after a BKZ reduction of blocksize β). Note that sometimes we have $\kappa \leq \beta$ in the optimal choice, which means that using BKZ only is more efficient than a two-step mode under this case.

4 Definition of Restricted SVP

4.1 Preparation: Restricted SVP with Fixed Length

We first give a simplified case for restricted SVP, and consider its solving algorithm. Although it is only a special case for our more generalized definition of restricted SVP, it makes good intuition for this new hard problem and how to solve it.

Intuitively, a solution for restricted SVP should satisfy two conditions: (1) it is a short vector in the lattice; (2) it can pass the restriction. This leads to our definition below:

Definition 4.1. *Given $len \in \mathbb{R}^+$, $\mathcal{R} : \mathbb{R}^d \mapsto \{0,1\}$ (called restriction), $\mathcal{L}$ is a d-dimensional lattice, the fixed-length restricted SVP (*$\mathrm{RSVP}^*(\mathcal{L}, len, \mathcal{R})$*) aims to find a vector $\mathbf{v} \in \mathcal{L} \cap \mathcal{B}^d(len)$ such that $\mathcal{R}(\mathbf{v}) = 1$.*

Let $p = \Pr(\mathcal{R}(\mathbf{v}) = 1 | \mathbf{v} \in \mathcal{L} \cap \mathcal{B}^d(len))$. (We note that for a normal predicate $\mathcal{R}$ with certain level of "uniformity", we can heuristically assume that $\Pr(\mathcal{R}(\mathbf{v}) | \|\mathbf{v}\| \leq len \wedge \mathbf{v} \in \mathcal{L}) \approx \Pr(\mathcal{R}(\mathbf{v}) | \|\mathbf{v}\| \leq len)$, which can be used to simplify the calculation of p.) Following the two-step solving algorithm, we directly get a trivial algorithm for solving fixed-length restricted SVP as follows:

input : Lattice $\mathcal{L}$ with basis $\mathbf{B}$, target RHF δ, sieving dimension κ;
output: A list of short lattice vectors S;

```
Function TrivialSolver(L, R, δ, κ):
    V ← TwoStepSolver(L, R, δ, κ);
    for each v ∈ V do
        if R(v) = 1 then
            return v;
    return "fail";
```

Algorithm 5: Trivial Restricted SVP Solver.

However, the success rate of this algorithm, which is $1 - (1-p)^{\sqrt{\frac{4}{3}}^{\kappa}}$ might be too large or too small for different values of p, which makes the algorithm inefficient in most cases. So how to solve a fixed-length restricted SVP with any given success rate?

In order to solve the problem with success rate r, we only need to find a set of vector which contains $S = \frac{\log(1-r)}{\log(1-p)}$ vectors in $\mathcal{L} \cap \mathcal{B}^d(len)$. If $S \leq \sqrt{4/3}^{\kappa}$, then the two-step mode alone is enough to solve the problem. However, if $S << \sqrt{4/3}^{\kappa}$, the generation of the large vector list seems to be wasteful. Is there a

more efficient algorithm which generates a list of less vectors than sieving with dimension κ? This is the first problem we wish to answer.

The other possibility is that, $S > \sqrt{4/3}^{\kappa}$, thus we cannot achieve the success rate r from the list of sieving vectors. Of course, since BKZ and sieving are randomized algorithms, we can run the algorithm with different seeds many times, until the success rate achieves r. However, such an algorithm is also wasteful, since we totally discard any intermediate results, including the reduced lattice basis from BKZ and the sieving list from the final sieve of dimension κ.

Another method suggested in the literature [33] is as follows: Let $\mathbf{B} = (\mathbf{b}_0, ..., \mathbf{b}_{d-1})$ be the reduced lattice basis after BKZ. Then each time we choose κ different vectors from $\mathbf{B}$, say $(\mathbf{b}_{i_0}, ..., \mathbf{b}_{i_{\kappa-1}})$, run lattice sieving on this sub-lattice to generate $\sqrt{4/3}^{\kappa}$ short vectors, repeat the procedure until we get enough short vectors, so the reduced basis from BKZ can be reused to generate more vectors. The main problem of this method is that, since the lengths of $\mathbf{b}_0, ..., \mathbf{b}_{d-1}$ are usually increasing, the shortest vector in $(\mathbf{b}_{i_0}, ..., \mathbf{b}_{i_{\kappa-1}})$ is also longer than the shortest vector in $(\mathbf{b}_0, ..., \mathbf{b}_{\kappa-1})$, which means the length of vectors generated using this method could be much longer than the length of vectors in the sieving list of lattice $\mathcal{L}[0:\kappa]$.

So the second problem we wish to solve is, how to find a method for generating a larger list of short vectors using only κ-dimensional sieve, while the length of vectors in this list is the same or with only slight increase than the length of vectors in the sieving list? We shall discuss the two problems in Section 5.

4.2 Restricted SVP with Related Probability

In our discussion above, we implicitly assume that the probability for a vector to pass the restriction is unrelated with its length. However, it is usually not the case. For example, if we let the restriction be $\|\mathbf{v}\|_\infty \leq B$, then the shorter $\mathbf{v}$ is, the higher probability it passes the restriction. Under this case, the restricted probability is related to the length of the lattice vector, which we consider it as a more general case.

Next, we formally define the problem.

Definition 4.2. *Given $\mathcal{R} : \mathbb{R}^d \mapsto \{0, 1\}$ (called restriction), $\mathcal{L}$ is a d-dimensional lattice, let $\mathcal{P}(len) = \Pr(\mathcal{R}(\mathbf{v}) = 1 | \mathbf{v} \in \mathcal{L} \wedge \|\mathbf{v}\| \leq len)$, we require that $\mathcal{P}(len) \geq \mathcal{P}(len')$ for any $len < len'$. The restricted SVP (*$\mathrm{RSVP}(\mathcal{L}, \mathcal{R})$*) aims to find a vector $\mathbf{v} \in \mathcal{L}$ such that $\mathcal{R}(\mathbf{v}) = 1$.*

We call $\mathcal{P}(len)$ the related probability function, which describes the relation between the restriction and the length of lattice vectors. Note that since we use a more general definition of $\mathcal{R}$, the bound of vector length is no longer taken as an input of the problem. If we require that the output vector length must below a certain value l, we only need to set $\mathcal{P}(len) = 0$ for $len > l$. But even there is no bound on the length of solution, since the shorter the vector is, the higher probability it passes the restriction, its solution is also a short lattice vector with

high probability, so our restricted SVP can still be considered as a variant of the standard (approximate) SVP.

Solving generalized restricted SVP is similar to solving the fixed-length version: we need to generate a set of short lattice vectors and check whether any vector in it can pass the restriction. However, since the probability that a vector passes the restriction increases as its length decreases, generating shorter vectors may increase the success probability. But since generating shorter vectors also takes more time, we should carefully choose the target length to minimize the time complexity of the solving algorithm.

Similar to Section 4.1 above, in most cases, we can heuristically assume that $\Pr(\mathcal{R}(\mathbf{v})|\|\mathbf{v}\| \leq len \wedge \mathbf{v} \in \mathcal{L}) \approx \Pr(\mathcal{R}(\mathbf{v})|\|\mathbf{v}\| \leq len)$ for any possible len, to simplify the calculation of $\mathcal{P}(len)$. However, in some cases (such as Kannan's embedding for approximate CVP, which we shall discuss in Section 6.1), the probability is defined and calculated only on lattice vectors.

5 Heuristic Algorithm for Restricted SVP

We already pointed out that restricted SVP can be solved by a two-step lattice solving algorithm which outputs a set of lattice vector. While sieving on κ-dimensional sub-lattice can output a set of size $\sqrt{4/3}^{\kappa}$, we need to construct more efficient algorithms for outputting a list of less or more short lattice vectors. We give two algorithms, called flexible-d4f and sieve-then-slice which output lists of vectors less/more than $\sqrt{4/3}^{\kappa}$ respectively. Then we combine the two algorithms to get a solving algorithm for restricted SVP with any parameters.

5.1 Flexible Dimensions for Free

For a lattice $\mathcal{L}$ of dimension d, we can perform lattice sieving either with or without dimension for free. If we perform sieving with f dimensions for free, the time cost is $\sqrt{3/2}^{d-f}$ which outputs one shortest lattice vector, and if we perform sieving with no dimension for free, the time cost is $\sqrt{3/2}^{d}$ which outputs $\sqrt{4/3}^{d}$ shortest lattice vectors.

So it is intuitive that if we perform the lattice sieving with f' dimensions for free for $0 < f' < f$, we can further balance between the time cost and the output size of shortest lattice vectors. Next, we present the algorithm with a more flexible dimensions for free in the searching step of a two-step approximate SVP solver, and give a theorem on the size of its output list.

Theorem 5.1. *Algorithm 6 returns a list of size approximately γ^{κ}.*

Proof. Let $L = \{\mathbf{v} : \mathbf{v} \in \mathcal{L} \wedge \|\mathbf{v}\| \leq \gamma \cdot gh(\mathcal{L}[0:\kappa])\}$ for $\gamma = \sqrt{4/3} \cdot \delta^{-\frac{f'd}{d-1}}$. From Gaussian Heuristic, we can see that there is $|L| \approx \gamma^{\kappa}$ as $\gamma \geq 1$.

Next, we show that for each vector $\mathbf{v} \in \mathcal{L}[0:\kappa]$ with $\|\mathbf{v}\| \leq \gamma \cdot gh(\mathcal{L}[0:\kappa])$, its projection $\pi_{f'}(\mathbf{v})$ is heuristically contained in the vector list generated from sieving on $\mathcal{L}[f':\kappa]$.

input : Lattice $\mathcal{L}$ with BKZ-reduced basis and RHF δ, sieving dimension κ, number of free dimensions f';
output: A list of short vectors $L \subset \mathcal{L}$;
Function `FlexibleD4F`$(\mathcal{L}, f')$**:**
  $L_{\text{sieve}} \leftarrow$ `Sieve`$(\mathcal{L}[f' : \kappa])$; $L \leftarrow \emptyset$;
  $\gamma \leftarrow \sqrt{4/3} \cdot \delta^{-\frac{f'd}{d-1}}$;
  for each $\mathbf{v} \in L_{\text{sieve}}$ **do**
    if $\|$`Babai-Lift`$(\mathbf{v})\| \leq \gamma \cdot gh(\mathcal{L}[0 : \kappa])$ **then**
      $L \leftarrow L \cup \{$`Babai-Lift`$(\mathbf{v})\}$;
  return L;

Algorithm 6: Flexible Dimension for Free

We take the same heuristic assumption as in [25], such that

$$\pi_{f'}(\mathbf{v}) \approx \sqrt{\frac{\kappa - f'}{\kappa}} \gamma \cdot gh(\mathcal{L}[0 : \kappa]),$$

so we only need to show that $\sqrt{\frac{\kappa-f'}{\kappa}}\gamma \cdot gh(\mathcal{L}[0 : \kappa]) \leq \sqrt{4/3} \cdot gh(\mathcal{L}[f' : \kappa])$.

Since the input lattice basis is BKZ-reduced, from Geometric Series Assumption:

$$gh(\mathcal{L}[0 : \kappa]) \approx \sqrt{\frac{\kappa}{2\pi e}} \left(\prod_{i=0}^{\kappa-1} \delta^d \alpha^i \right)^{1/\kappa} = \sqrt{\frac{\kappa}{2\pi e}} \delta^d \alpha^{\frac{\kappa-1}{2}}.$$

Similarly, we have:

$$gh(\mathcal{L}[f' : \kappa]) \approx \sqrt{\frac{\kappa - f'}{2\pi e}} \delta^d \alpha^{\frac{f'+\kappa-1}{2}}.$$

Thus we only need to show that $\gamma \leq \sqrt{4/3} \cdot \alpha^{f'/2}$, which is promised by $\gamma = \sqrt{4/3} \cdot \delta^{-\frac{f'd}{d-1}}$ since $\alpha = \delta^{-\frac{2d}{d-1}}$ from Geometric Series Assumption. □

Note. (1) In the algorithm above, we heuristically assume that every vector $\mathbf{v} = \sum_{i=0}^{d-1} v_i^* \mathbf{b}_i^*$ of length $\gamma \cdot gh(\mathcal{L}[0 : \kappa])$ can be recovered from Babai's lifting, which means that we should have $\frac{1}{2}\|\mathbf{b}_i^*\| \geq v_i^*$ for $i = 0, ..., f' - 1$. This condition can be satisfied for most vectors, as v_i^* has a high probability around:

$$\frac{1}{\sqrt{\kappa}} \gamma \cdot gh(\mathcal{L}[0 : \kappa]) = \sqrt{\frac{2}{3\pi e}} \delta^d \alpha^{\frac{\kappa-1+f'}{2}} < 0.28 \alpha^{\frac{\kappa-f'}{2}} \|\mathbf{b}_{f'-1}^*\| \leq 0.28 \alpha^{\frac{\kappa-f'}{2}} \|\mathbf{b}_i^*\|,$$

which is much smaller than $0.5\|\mathbf{b}_i^*\|$, since $\alpha < 1$ and $\kappa > f'$.

(2) In Algorithm 6, we only output a list of vector which lengths $\leq len$. However, for our generalized version of restricted SVP defined in Section 4.2, we can see that even if the lifted vector has length $> len$, there is still a probability that it could pass the restriction. So we in fact overestimated the hardness of

restricted SVP if we use Algorithm 6 for its complexity. But since our work only aims to give an heuristic solving algorithm for restricted SVP, we leave the problem of presenting a tighter hardness bound of this problem into our future work.

We tested the algorithm with sieving dimension $\kappa = 70$ and number of free dimensions $f' = 0, ..., 15$, using the implementation of lattice sieving in G6K [5]. The size of the output list is shown in the figure below:

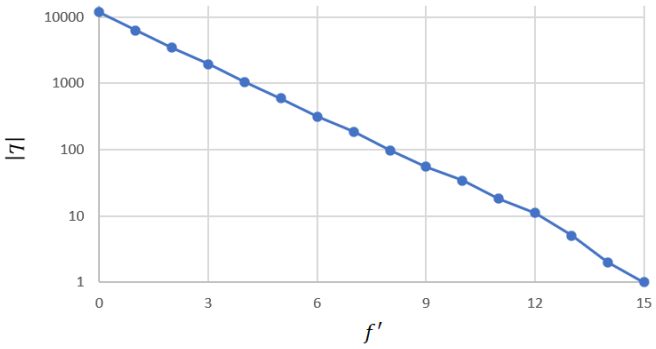

Fig. 1. Number of vectors in the output of flexible d4f with $\kappa = 70$.

5.2 The Sieve-then-Slice Technique

In [23], the authors suggested that the randomized slicer algorithm can be used to accelerate the solving of (exact) SVP, and further discussed in [24]. Their method, we abbreviate it as the enumeration-then-slice technique, is as follows:

For a lattice with dimension d, divide the lattice into two sub-lattices, say $\mathcal{L}^{\perp} = \mathcal{L}(\mathbf{b}_0, ..., \mathbf{b}_{\kappa-1})$ and $\mathcal{L}^{\top} = \mathcal{L}(\mathbf{b}_{\kappa}, ..., \mathbf{b}_{d-1})$. To solve SVP on $\mathcal{L}$, first enumerate a set of lattice vectors T on $\mathcal{L}^{\top}$, then for each $\mathbf{t} \in T$, find its closest vector $\mathbf{w}$ in $\mathcal{L}^{\perp}$, until $\mathbf{w} - \mathbf{t}$ is the shortest vector in $\mathcal{L}$.

However, this method does not behave well enough in practice. The reason is not hard to understand: for any lattice basis $\mathbf{B} = (\mathbf{b}_0, ..., \mathbf{b}_{d-1})$ and the shortest vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$, if we write $\mathbf{v} = \sum_{i=0}^{d-1} c_i \mathbf{b}_i$, there is no promise that any "part" of it, say $\mathbf{v}^{\top} = \sum_{i=\kappa}^{d-1} c_i \mathbf{b}_i$ is also short. In fact, there is a high probability that $\mathbf{v}^{\top}$ is outside the enumeration range of $\mathcal{L}^{\top}$, so the success rate of this algorithm should be rather small, especially for large dimension d.

However, for solving restricted SVP, our goal is only to generate a list of short enough lattice vectors, instead of finding a specific shortest vector. It is not hard to see that the enumerate-then-slice technique fits well for such case.

To simplify our further discussion, we let $\mathcal{L}^{\top} = \mathcal{L}[\kappa : d]$ be the projected sub-lattice, instead of the sub-lattice in [24]. Also, we replace enumeration by sieving to get a more accurate size for the output vector list. We note that

enumeration may be faster than sieving for small dimensions, but since the time cost for generating short vectors on $\mathcal{L}^\top$ is only a small fraction of the total time cost, changing the sub-procedure here has its impact small enough that could be ignored. We call our algorithm the sieve-then-slice algorithm.

Before presenting the algorithm, we define the following Lift operation: For each $\mathbf{v} = \sum_{i=\kappa}^{d-1} c_i \pi_\kappa(\mathbf{b}_i) \in \mathcal{L}[\kappa : d]$, $\mathsf{Lift}(\mathbf{v}) = \sum_{i=\kappa}^{d-1} c_i \mathbf{b}_i \in \mathcal{L}$.

We give the detailed algorithm of our sieve-then-slice approach for generating a list of short lattice vectors.

input : Lattice $\mathcal{L}$ with RHF δ, sieving dimension κ, size of output list $S > \sqrt{4/3}^\kappa$;
output: A list of short vectors $L \subset \mathcal{L}$ with $|L| \approx S$;

Function `Sieve-Then-Slice`$(\mathcal{L}, \kappa, S)$**:**
- $L_{\text{sieve}} \leftarrow$ `Sieve`$(\mathcal{L}([0:\kappa]))$;
- $\varphi \leftarrow \lceil \log(\frac{S - \sqrt{4/3}^\kappa}{\sqrt{16/13}^\kappa}) / \log \sqrt{4/3} \rceil + C$; // $C = \omega(1)$
- $T' \leftarrow$ `Sieve`$(\mathcal{L}[\kappa : \kappa + \varphi])$; $T \leftarrow \emptyset$;
- **for each** $\mathbf{t}' \in T'$ **do**
  - $T \leftarrow T \cup \{$`Lift`$(\mathbf{t}')\}$;
- **return** `ModifiedRandomizedSlicer` (T, L_{sieve}, S);

Algorithm 7: Sieve-Then-Slice

Theorem 5.2. *Each vector in the output list of Algorithm 7 has its length heuristically* $\leq \sqrt{\frac{4}{3}(gh(\mathcal{L}[0:\kappa])^2 + \|\mathbf{b}_\kappa^*\|^2)}$.

Proof. Before we go into further details, we first point out that, the analysis for randomized slicer [23, 26] cannot be directly applied to T in our algorithm. In [23, 26], only full-rank lattices are considered, which means that for a d-dimensional lattice, the target vector should be in a same d-dimensional vector space with all lattice vectors.

To handle such case, we use a method similar to the co-lattice algorithm [28] as follows: For each vector $\mathbf{t} \in \mathbb{R}^d$, we break it into two orthogonal vectors: $\mathbf{t} = \mathbf{t}^\perp + \mathbf{t}^\top$, where $\mathbf{t}^\perp$ is in the same κ-dimensional subspace with $\mathcal{L}[0:\kappa]$, and $\mathbf{t}^\top$ is orthogonal with $\mathcal{L}[0:\kappa]$. By the definition of orthogonal basis $\mathbf{B}^*$ and the projection operator π_κ, we can see that $\mathbf{t}^\top = \pi_\kappa(\mathbf{t}) \in \mathcal{L}[\kappa : d]$. For each $\mathbf{t} \in T$, $\mathbf{t}$ is generated from $\mathbf{t}'$ by performing Lift, we can see that $\mathbf{t}^\top = \mathbf{t}'$ is a vector generated from lattice sieving on $\mathcal{L}[\kappa : \kappa+\varphi]$. Suppose that $\varphi \leq \beta$ with $\delta = \delta(\beta)$, we can see that $\mathbf{b}_\kappa^*$ is the shortest non-zero vector in $\mathcal{L}[\kappa : \kappa+\varphi]$ by the property of BKZ-β reduced basis, so $\|\mathbf{t}'\| \leq \sqrt{4/3}\|\mathbf{b}_\kappa^*\|$.

It directly follows that performing randomized slicer on $\mathbf{t}$ does not change $\mathbf{t}^\top$, only $\mathbf{t}^\perp$. So for each vector $\mathbf{v} \in S$ generated from randomized slicing on $\mathbf{t}$, we have that $\mathbf{v}^\top = \mathbf{t}^\top$, which has length $\leq \sqrt{4/3}\|\mathbf{b}_\kappa^*\|$.

For the length of $\mathbf{v}^\perp$, which follows the actual output of the iterative slicer, we refer to Lemma 10 in [23], which claims that if the slicer uses a list of size

$\alpha^{d+o(d)}$, the output of the slicer after each iteration has its length $\leq \beta \cdot \mathcal{L}$ with $\beta = \alpha^2/2\sqrt{\alpha^2-1}$. For $\alpha = \sqrt{4/3}$ which corresponds with lattice sieving, we have that $\beta = \sqrt{4/3}$, which means that $\|\mathbf{v}^{\perp}\| \leq \sqrt{4/3}gh(\mathcal{L}[0:\kappa])$.

So the length of the output vector $\mathbf{v}$ satisfies:

$$\|\mathbf{v}\| = \sqrt{\|\mathbf{v}^{\perp}\|^2 + \|\mathbf{v}^{\top}\|^2} \leq \sqrt{\frac{4}{3}(gh(\mathcal{L}[0:\kappa])^2 + \|\mathbf{b}_{\kappa}^*\|^2)}.$$

□

Our experiment shows that almost all ($> 99.5\%$) output vectors have their length inside this bound. In the sieve-then-slice approach, the sieving dimension κ might be smaller than the BKZ dimension β for $\delta = \delta(\beta)$, but it cannot occur in the flexible d4f algorithm.

Note. (1) There is a small probability that the output of randomized slicer collides with each other: a same vector is outputted after different iterations. This does affect the efficiency for solving CVP, but in our algorithm, it causes the size of output list $|L|$ smaller than the number of iterations S. To avoid this problem, we increase φ by $C = \omega(1)$, so the size of T becomes $\sqrt{4/3}^{C}$ times larger than the batch size as in [26].

For each vector $\mathbf{t} \in T$, it is heuristically assumed in [26] that each iteration randomly outputs a vector from a list size of $S' = \sqrt{16/13}^{\kappa+o(\kappa)}$. By increasing the size of T, the number of iterations for each $\mathbf{t} \in T$ also becomes $S'/\sqrt{4/3}^{C}$, thus we can bound the collision probability by $\sum_{i=0}^{S'/\sqrt{4/3}^{C}} \frac{i}{S'} \approx \frac{1}{2\sqrt{4/3}^{C}}$. So by letting $C = \omega(1)$, we have that the collision probability could be asymptotically ignored.

We note that by choosing a larger list T, the number of iterations for each $\mathbf{t} \in T$ becomes insufficient to output the CVP solution for every $\mathbf{t}$. But it does not really matter, since our aim is only to output a list of short vectors in $T + \mathcal{L}$, not the closest vectors.

(2) We note that like in Section 5.1, we also overestimated the time complexity of restricted SVP from Algorithm 7. The reason is that the output list L cannot be considered as uniform in $\mathcal{B}^{\kappa}(r)$ for $r = \sqrt{\frac{4}{3}(gh(\mathcal{L}[0:\kappa])^2 + \|\mathbf{b}_{\kappa}^*\|^2)}$: for the sieving list, the vectors has lengths $\leq \frac{4}{3}gh(\mathcal{L}[0:\kappa]) < r$, and for the slicer list, the first part $\mathbf{v}^{\perp}$ of $\mathbf{v}$ is chosen from an approximate Voronoi cell $\mathcal{V}$ which contains $\sqrt{16/13}^{d}$ possible vectors. Since $\mathcal{V}$ is a convex shape contained in $\mathcal{B}^d\left(\sqrt{\frac{4}{3}}gh(\mathcal{L}[0:\kappa])\right)$, $\mathbf{v}^{\perp}$ is not uniform in $\mathcal{B}^d\left(\sqrt{\frac{4}{3}}gh(\mathcal{L}[0:\kappa])\right)$, but the possibility for $\mathbf{v}^{\perp} \leq \sqrt{\frac{16}{13}}gh(\mathcal{L}[0:\kappa])$ is higher than $\mathbf{v}^{\perp} > \sqrt{\frac{16}{13}}gh(\mathcal{L}[0:\kappa])$. So we also overestimated the length of $\mathbf{v}^{\perp}$, hence the whole vector $\mathbf{v}$.

By the discussion above, the list is denser for smaller vector lengths, which means that $|L \cap \mathcal{B}^{\kappa}(r')| > |L| \cdot (\frac{r'}{r})^{\kappa}$ for some $r' < r$. Since we assume that the related probability $\mathcal{R}(r') > \mathcal{R}(r)$, the probability that L passes the restriction

is also higher than our estimation. As in Section 5.1, we leave the problem of presenting a tighter bound to our future work.

(3) If we combine the sieve on $\mathcal{L}[\kappa : \kappa+\varphi]$ with flexible dimension for free in Section 5.1, we can get a list of shorter vectors T with larger sieving cost. It is unclear whether such modification will increase or decrease the time complexity for restricted SVP. But since both lengths of vectors in T and the sieving cost on $\mathcal{L}[\kappa : \kappa+\varphi]$ are small compared with the lengths of vectors in L and the total time cost of Algorithm 7, we can see that such modification has only small affect on the time complexity, so we simply ignore the possible use of dimension for free on $\mathcal{L}[\kappa : \kappa+\varphi]$.

(4) We already mentioned that the sieve-then-slice approach should be used when the probability $\mathcal{P}(len)$ is smaller than the inverse of the size of sieving list $\sqrt{4/3}^{-\kappa}$. However, if $\mathcal{P}(len)$ is exponentially smaller than $\sqrt{4/3}^{-\kappa}$, the sieve-then-slice approach also becomes inefficient. Under such case, we can first generate a lattice basis with lengths $O(len/\sqrt{d})$, then use Gaussian sampling to generate enough short vectors, as Gaussian samplers can be done in polynomial time. However, such cases are outside the range of this paper.

5.3 Algorithm for Solving Restricted SVP

Finally, we combine the two approaches together, to give our solving algorithm for restricted SVP in Algorithm 8.

input : Lattice $\mathcal{L}$, target RHF δ, sieving dimension κ, restriction $\mathcal{R}$ with related probability function $\mathcal{P}$, success rate p;
output: A vector $\mathbf{v} \in \mathcal{L}$ such that $\mathcal{R}(\mathbf{v}) = 1$ with success rate p;

Function `RestrictedSVPSolver`$(\mathcal{L}, \delta, \kappa, \mathcal{R}, p)$**:**
  $\mathbf{B} \leftarrow$ `BKZ`$(\mathcal{L}, \delta)$;
  $len \leftarrow \sqrt{4/3} \cdot \sqrt{gh(\mathcal{L}(\mathbf{B}[0:\kappa]))^2 + \|\mathbf{b}_\kappa^*\|^2}$;
  $size \leftarrow \frac{\log(1-p)}{\log(1-\mathcal{P}(len))}$;
  if $size > (\sqrt{4/3})^\kappa$ **then**
    $L \leftarrow$ `Sieve-Then-Slice`$(\mathcal{L}, \kappa, size)$
  else
    $len \leftarrow \sqrt{4/3} \cdot gh(\mathcal{L}(\mathbf{B}([0:\kappa]))$;
    $size \leftarrow \sqrt{4/3}^\kappa$; $f' = 0$;
    while $size \geq \frac{\log(1-p)}{\log(1-\mathcal{P}(len))}$ **do**
      $f' \leftarrow f' + 1$; $\gamma = \sqrt{4/3} \cdot \delta^{-\frac{f'd}{d-1}}$;
      $len \leftarrow \gamma \cdot gh(\mathcal{L}[0:\kappa])$;
      $size \leftarrow \gamma^\kappa$;
    $L \leftarrow$ `FlexibleD4F`$(\mathcal{L}, \kappa, f'-1)$;
  Search through L to find $\mathbf{v} \in L$ such that $\mathcal{R}(\mathbf{v}) = 1$;
  return $\mathbf{v}$;

Algorithm 8: Solving Restricted SVP

Time cost model and optimal parameter choices. We note that in Algorithm 8, we take δ, κ as input parameters. To actually solve a restricted SVP sample, we need to find the optimal δ, κ which minimize the time cost. This requires a concrete time cost model for BKZ, lattice sieving and randomized slicer.

We assume that the calculation of the predicate $\mathcal{R}(\mathbf{v})$ costs only polynomial time. Since generating each vector $\mathbf{v}$ costs exponential time in average (see our discussion below), we can ignore the time cost of $\mathcal{R}$, only consider the time cost of lattice algorithms.

Next, we give a discussion on which time-cost model we should use. There are different ways to determine the time cost model in the literature:

(1) Asymptotic complexity. It is well known that the most efficient lattice sieve BDGL [16] costs $\sqrt{3/2}^{n+o(n)}$ for lattice dimension n, where $n = \kappa$ for the sieve-then-slice approach, and $n = \kappa - f'$ in the flexible-d4f approach. For the cost of randomized slicer in the sieve-then-slice approach, we use the result in [23,26], which claims that after each iteration, a vector is generated by traversing through the nearest neighbor search (NNS) data structure which costs $\sqrt{9/8}^{\kappa+o(\kappa)}$. In our modified version of randomized slicer, we add every vector from each iteration into the output list L_{out}, instead of only finding the closest vector as in the original version. So the time cost of output a list of size S is approximately $S \cdot \sqrt{9/8}^{\kappa+o(\kappa)}$. Under this approximation, outputting a list of $\sqrt{4/3}^{\kappa+o(\kappa)}$ vectors through randomized slicer has time cost $\sqrt{3/2}^{\kappa+o(\kappa)}$, which is exactly the same as lattice sieving. So for generating $S > \sqrt{4/3}^{\kappa+o(\kappa)}$ lattice vectors in the sieve-then-slice approach, the total cost of lattice sieving and randomized slicer is asymptotically $S \cdot \sqrt{9/8}^{\kappa+o(\kappa)}$.

By the discussion in [8], the cost of BKZ can be considered as a sum of the costs of polynomial number of calls to lattice sieving with dimension β for $\delta = \delta(\beta)$. Ignoring the polynomial factors, we simply let the cost of BKZ to be $\sqrt{3/2}^{\beta+o(\beta)}$, which is called the core-SVP model for BKZ time cost. Let $\alpha = \max\{\kappa, \beta\}$, the total time cost is asymptotically $\sqrt{3/2}^{\alpha+o(\alpha)}$. However, since asymptotic complexity ignores constant factors (also polynomial factors in BKZ), it cannot be used to optimize the choice of δ and κ. We must use other methods.

(2) Gate-count model. In [7], the authors estimated the gate count of the BDGL sieve [16], which are further used to estimate a concrete hardness for LWE [22]. The hardness of LWE from gate-count model is generated from summing up all costs of lattice sieving in a progressive BKZ, while considering both progressive sieve and dimension for free.

A main obstacle for presenting a gate-count for our algorithm, is that there is currently no gate-count model for randomized slicer. But since both lattice sieving and randomized slicer use the same data structure NNS, and the gate-count for lattice sieving is calculated from the average number of visits to the data structure, we can heuristically assume that visiting NNS in both sieving and slicer cost a same number of gates, thus generating $\sqrt{4/3}^{d}$ vectors from

randomized slicer has a same gate count as generating $\sqrt{4/3}^d$ vectors from lattice sieving, which is the gate count of a d-dimensional sieve.

(3) Practical time cost. Here practical time cost means the time cost of the algorithm on a certain computer, recorded in CPU cycles, seconds, etc. To generate a practical time cost model, the solving algorithm is first tested through experiments with different parameters, then use statistical methods such as least square fits to find a relation between the time cost and the problem size, as shown in [5,27]. Below is the experimental result for the average time cost of both lattice sieving and (modified) randomized slicer using G6K for outputting one vector with different κ.

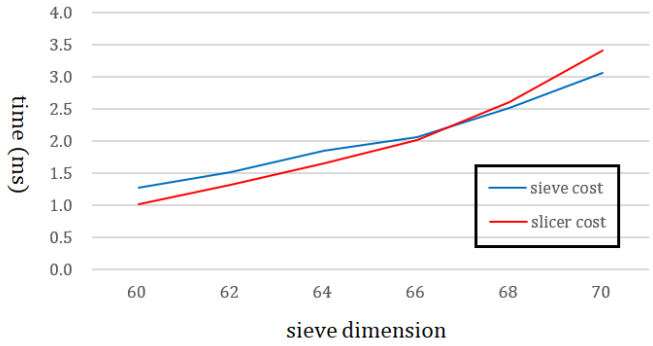

Fig. 2. Comparison between the average time of outputting one vector using sieve and slicer. Times are in milliseconds.

We can see that the two costs are quite close with each other. However, we observe an interesting crossover, which suggests that the time cost of slicer may grow faster than sieving. We leave a more detailed analysis in our future work.

Practical time cost model should be used if we aim to actually solve an instance of restricted SVP. However, if our goal is to estimate the security level of a lattice-based cryptosystem based on restricted SVP, we should use the gate-count model instead, since we cannot run experiments on parameters of cryptographic level (2^{128} or more). Since practical time cost is highly related with the implementation of algorithms, sometimes even the hardware of the system, we do not give a more detailed discussion here.

We note that after given the time cost model, Algorithm 8 implies an estimator for the time complexity of solving restricted SVP with parameter δ, κ. So we only need to search through all possible δ, κ to find the optimal parameter choices, and get the heuristic hardness estimation of restricted SVP without actually running the solving algorithm on the lattice $\mathcal{L}$ (the length of vectors in $\mathbf{B}^*$ can be estimated from GSA).

Why not combine the two? There is a natural question that why we do not combine the two techniques instead of using only one at a time. To explain the

reason, we change our point of view on Algorithm 8: we consider the parameter choice as a dynamic procedure. The starting point is performing lattice sieve on $\mathcal{L}([0:\kappa])$ with full dimension for free f without using sieve-then-slice, which returns a list of $O(1)$ lattice vectors.

To get a list of S vectors, we use two methods in changing the parameters to increase the size of output list: the first one is decreasing the number of free dimensions f; the second one is increasing the number of vectors in T, so the sieve-then-slice approach can generate more vectors before dimension for free, thus resulting in a larger output list.

We can see that the first approach is more efficient than the second: decreasing f by 1 results in a constant $(\sqrt{3/2})$ times increase in the time cost and $\delta^{\frac{\kappa d}{d-1}} = \Omega(\sqrt{\beta})$ times increase in the output size for $\delta = \delta(\beta)$ if $\kappa > \beta$, while the growth speed in time complexity and output size are the same in the second approach. The only cost for the first approach, is that the lengths of output vectors are also increased by $\delta^{\frac{d}{d-1}}$ times, but it cannot cancel out the time cost, unless the related probability function $\mathcal{P}(len)$ decreases exponentially in κ as len increases, which is quite uncommon that could be ignored.

By our discussion, we claim that using one technique at a time is always better than combining the two, and this is why we design Algorithm 8 as above.

6 Application of Restricted SVP

6.1 Restriction Probability for Kannan's Embedding

Let $\mathcal{L}$ be a d-dimensional lattice with basis $\mathbf{B}$. To solve CVP with target vector $\mathbf{t}$ on lattice $\mathcal{L}$, we can construct a new lattice $\mathcal{L}' = \mathcal{L}\begin{pmatrix} \mathbf{B} & \mathbf{t} \\ \mathbf{0}^T & M \end{pmatrix}$ which embeds both $\mathcal{L}$ and $\mathbf{t}$, then solve SVP on the embedded lattice.

We have already shown that Kannan's embedding turns approximate CVP into the problem of restricted SVP: $\mathcal{R}(\mathbf{v}') = 1$ if and only if the last element in $\mathbf{v}'$ (say, v'_d) is $\pm M$. To further apply our heuristic algorithm in Section 5.3 for solving approximate CVP, we only need to calculate the related probability function $\mathcal{P}$.

The embedding parameter M is important in solving approximate CVP. If we choose an M too large, then the probability that $v'_d = 0$ might be too high, but if we choose an M too small, then the probability that $|v'_d| > 1$ might be too high. So to maximize the probability $\mathcal{P}(len' = \sqrt{len^2 + M^2})$ for the target length len we require, we should also find a way to optimize M.

Next, we give a result on the value of $\mathcal{P}$, as well as an heuristic choice for M related to len.

Theorem 6.1. *$\mathcal{P}(len) > 6 - 4\sqrt{2} \approx 0.343$ for appropriately chosen M.*

Proof. Each vector in $\mathcal{L}'$ must has its last element of the form cM by its construction. For each $c \in \mathbb{Z}$, suppose that $\mathbf{v}' = \binom{\mathbf{v}}{cM} \in \mathcal{L}'$ and $\|\mathbf{v}'\| \leq len'$, then $\|\mathbf{v}\| \leq \sqrt{len'^2 - c^2M^2}$.

Moreover, we can express $\mathbf{v}' = \sum_{i=0}^{d-1} c_i \mathbf{b}'_i + c \cdot \binom{\mathbf{t}}{M}$, and $\mathbf{b}'_i = \binom{\mathbf{b}_i}{0}$, so $\mathbf{v}$ combined by the first d elements of $\mathbf{v}'$ has: $\mathbf{v} - c\mathbf{t} = \sum_{i=0}^{d-1} c_i \mathbf{b}_i \in \mathcal{L}$.

So $\mathbf{v} - c\mathbf{t}$ is a lattice vector contained in a d-dimensional ball with radius $\sqrt{len'^2 - c^2M^2}$ and center $c\mathbf{t}$, which heuristically contains $\left(\frac{\sqrt{len'^2 - c^2M^2}}{gh(\mathcal{L})}\right)^d$ elements assuming Gaussian Heuristic, which means that the set $S'_c = \{\mathbf{v}' \in \mathcal{L}' : \|\mathbf{v}'\| \leq len' \wedge v'_d = cM\}$ has size $|S'_c| \approx \left(\frac{\sqrt{len'^2 - c^2M^2}}{gh(\mathcal{L})}\right)^d$. Since $\lim_{x\to\infty}(1-\frac{1}{x})^x = 1/e$, we heuristically replace $(len'^2 - c^2M^2)$ by $len'^2 e^{-\frac{c^2M^2}{len'^2}}$, then:

$$\left(\frac{\sqrt{len'^2 - c^2M^2}}{gh(\mathcal{L})}\right)^d \approx (len' \cdot gh(\mathcal{L}))^d \cdot e^{-\frac{dc^2M^2}{2len'^2}}.$$

Write $x = e^{-\frac{dM^2}{2len'^2}}$, we have $|S'_c| \approx (len' \cdot gh(\mathcal{L}))^d \cdot x^{c^2}$.

We can represent: $\mathcal{P}(len') = \frac{|S'_1|+|S'_{-1}|}{\sum_{c\in\mathbb{Z}}|S'_c|}$, thus we have:

$$\mathcal{P}(len') \approx \frac{2x}{1+2\sum_{i=1}^{\infty} x^{i^2}} > \frac{2x}{1+2\sum_{i=1}^{\infty} x^{i}} = \frac{2x}{\frac{2}{1-x}-1} = \frac{2x(1-x)}{1+x}.$$

The function $\left(\frac{2x(1-x)}{1+x}\right)' = 0$ has only one solution in $(0,1)$, which is $x = \sqrt{2}-1$, we can check that the maximal value of $\frac{2x(1-x)}{1+x}$ is $6-4\sqrt{2} \approx 0.343$ where $x = \sqrt{2}-1$. We can further calculate that

$$M = len' \cdot \sqrt{\frac{2}{d}} \ln(\sqrt{2}+1) \approx 1.24\frac{len'}{\sqrt{d}}.$$

□

Next, we test the probability that a short vector in the embedded lattice passes the restriction. We do the following experiment: first construct a random d-dimensional lattice $\mathcal{L}$ and a random target vector, use Kannan's embedding to generate a lattice $\mathcal{L}'$ with dimension $d+1$, then perform sieving on lattice $\mathcal{L}'$. We show the probability that an output vectors in the sieving list has its last element $\pm M$ for d from 60 to 70, which can be considered as the success probability of approximate CVP with $\gamma = \sqrt{4/3} \approx 1.15$.

We can see from Figure 3 that the probability has large deviation from our theoretical result, which is ≈ 0.343. This is because that in the proof of Theorem 6.1, we replace $(len'^2 - c^2M^2)$ by $len'^2 e^{-\frac{c^2M^2}{len'^2}}$, which is only valid when $\frac{c^2M^2}{len'^2} = O(1/d)$ is small, but we cannot say so for d from 60 to 70. We claim that for larger dimension d, the probability will finally converges to our theoretical result.

However, our experiment fails when we use the two-step solving algorithm on a lattice with larger dimension. Due to some reason that we failed to explain at the time, the LLL reduction tends to move the basis vector with last element

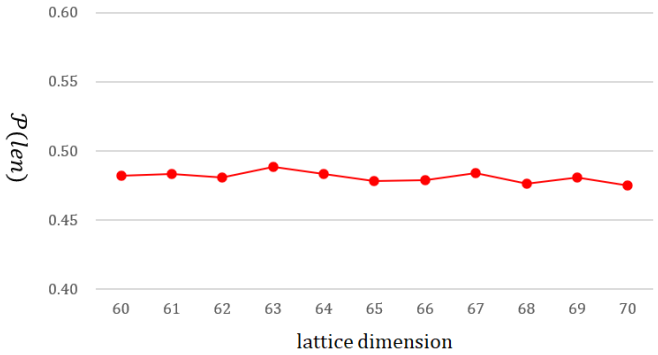

Fig. 3. The success probability of Kannan's embedding.

non-zero into the last few places in the basis, so these vectors rarely take part in the BKZ reduction, which results in the following phenomena: the first κ basis elements have their last elements all zero, which means that we shall never get a lattice vector with last element non-zero from sieving in $\mathcal{L}[0:\kappa]$. We are still working to find out the reason and fix the experiment.

6.2 Restriction Probability for SVP with Infinity Norm

Let B be the bound of the infinity norm, $\mathcal{R}_B(\mathbf{v}) = 1$ if and only if $\|\mathbf{v}\|_\infty \leq B$. We write $\mathcal{P}_B(len)$ be the related probability function. We assume that $\mathcal{P}_B(len) = \Pr(\|\mathbf{v}\|_\infty \leq B | \mathbf{v} \in \mathcal{L} \wedge \|\mathbf{v}\| \leq len) \approx \Pr(\|\mathbf{v}\|_\infty \leq B | \|\mathbf{v}\| \leq len)$.

We give the following estimation, which is also used in the security estimation of Dilithium [13], Section C.3, but without formal proof:

Theorem 6.2. *(1) Uniform distribution on $\mathcal{B}^d(len)$ is close to a d-dimensional spherical Gaussian $\mathcal{N}\left(\mathbf{0}, \frac{len}{\sqrt{d}} \cdot \mathbf{I}\right)$.*

(2) The probability of $\|\mathbf{v}\|_\infty \leq B$ for $\mathbf{v}$ uniform in $\mathcal{L} \cap \mathcal{B}^d(len)$ is approximately $\left(1 - 2\Phi\left(-\frac{\sqrt{d} \cdot B}{len}\right)\right)^d$, where Φ is the cumulated distribution function for the standard Gaussian $\mathcal{N}(0,1)$.

Proof. (1) We first show that each element in a vector uniform in $\mathcal{B}^d(len)$ is close to the distribution $\mathcal{N}(0, len/\sqrt{d})$, then show that these elements are approximately independent with each other.

We can see that, $\{\mathbf{v} : \mathbf{v} \in \mathcal{B}^d(len) \wedge v_i = x\}$ for $x \leq len$ is a $d-1$-dimensional ball with radius $\sqrt{len^2 - x^2}$. Use the approximate function for volume of high dimensional balls:

$$\mathrm{vol}(\mathcal{B}^d(len)) \approx \frac{1}{\sqrt{2\pi}} \left(\sqrt{\frac{2\pi e}{d}} len\right)^d,$$

we can write the probability function of v_i as:

$$\frac{\mathrm{vol}(\mathcal{B}^{d-1}(\sqrt{len^2-x^2}))}{\mathrm{vol}(\mathcal{B}^d(len))} \approx \sqrt{\frac{d}{2\pi e}}\left(\frac{d}{d-1}\right)^{(d-1)/2} \cdot \frac{\sqrt{len^2-x^2}^{d-1}}{len^d}.$$

Use the approximation: $(1+\frac{1}{x})^x \approx e$ for large enough x, we have that:

$$\left(\frac{d}{d-1}\right)^{(d-1)/2} = \sqrt{\left(1+\frac{1}{d-1}\right)^{d-1}} \approx \sqrt{e},$$

also:

$$\left(\frac{\sqrt{len^2-x^2}}{len}\right)^{d-1} = \left(\frac{len^2-x^2}{len^2}\right)^{\frac{d-1}{2}} = \left(\left(1-\frac{x^2}{len^2}\right)^{-\frac{len^2}{x^2}}\right)^{-\frac{(d-1)x^2}{2len^2}} \approx e^{-\frac{d\cdot x^2}{2len^2}}.$$

Set $\sigma = len/\sqrt{d}$, the probability function of v_i approximately equals to:

$$\frac{\sqrt{d}}{len}\frac{1}{\sqrt{2\pi}}e^{-\frac{d\cdot x^2}{2len^2}} = \frac{1}{\sqrt{2\pi}\cdot\sigma}e^{-\frac{x^2}{2\sigma^2}},$$

which is exactly the distribution of $\mathcal{N}(0,\sigma)$.

Now, we calculate the probability function for $(v_i = x_i, v_j = x_j)$, We can see that $\{\mathbf{v} : \mathbf{v} \in \mathcal{B}^d(len) \wedge v_i = x_i \wedge v_j = x_j\}$ is a $d-2$-dimensional ball with radius $\sqrt{len^2 - x_i^2 - x_j^2}$. Using a discussion similar to above, the probability function $\mathrm{vol}\left(\mathcal{B}^{d-2}\left(\sqrt{len^2-x_i^2-x_j^2}\right)\right)/\mathrm{vol}(\mathcal{B}^d(len))$ can be approximately calculated as $\frac{1}{2\pi\sigma^2}e^{-\frac{x_i^2+y_i^2}{2\sigma^2}}$ which is the product of the probability $v_i = x_i$ and $v_j = x_j$. So v_i and v_j are approximately independent, which means that the distribution can be approximated by the Gaussian distribution $\mathcal{N}\left(\mathbf{0}, \frac{len}{\sqrt{d}}\cdot\mathbf{I}\right)$.

(2) Since $\mathcal{N}(0,\sigma)(x) = \mathcal{N}(0,1)(x/\sigma)$, the probability that $|v_i| \leq B$ can be approximated by:

$$\int_{-B/\sigma}^{B/\sigma} \mathcal{N}(0,1)(x)\mathrm{d}x = \Phi(B/\sigma) - \Phi(-B/\sigma) = 1 - 2\Phi(-B/\sigma).$$

□

The approximation is only valid when v_i is small compared with len, since B is the bound of v_i, it also means that B must be small compared with len. We note that this requirement can be easily satisfied, since in order to optimize the time cost in the restricted SVP solver, the output length of the set of lattice vectors cannot be too large. We omit the detailed discussion here, but shall give an example later in Section 6.3.

We show that this approximation is accurate by experiments as follows: sample k uniform random vectors on the ball $\mathcal{B}^d(len)$, return l/k where l is the number of vectors which infinity norm $\leq B$.

We further explain how to sample a random vector on $\mathcal{B}^d(len)$: we make use of the fact that a high-dimensional Gaussian is a spherical distribution. We first sample $a_0, ..., a_{d-1} \sim \mathcal{N}(0,1)$, $b \sim \mathcal{U}[0,1]$, let $\mathbf{a} = (a_0, ..., a_{d-1})^T$, return $\mathbf{v} = len \cdot b^{1/d}\mathbf{a}/\|\mathbf{a}\|$.

The factor $len \cdot b^{1/d}$ is used to make the length of $\mathbf{v}$ follows the same distribution from the length of a uniform vector in $\mathcal{B}^d(len)$: note that the probability for a uniform point in $\mathcal{B}^d(len)$ has length $\leq len \cdot b^{1/d} = x$ has probability: $x^d/len^d = b$.

The experiment result is shown in the figure below, where the estimated probability is close to the real distribution as the two curves almost overlap.

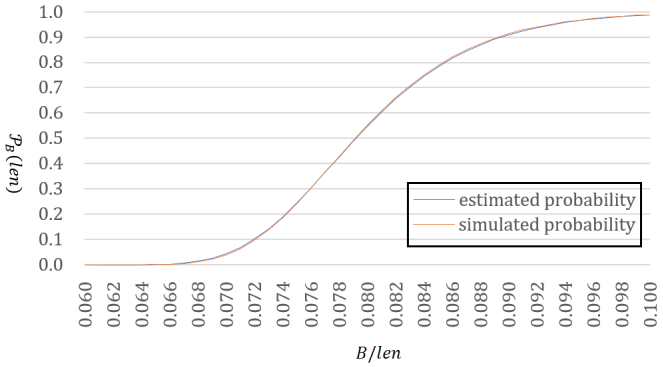

Fig. 4. Comparison between estimated and simulated probability for $\mathcal{P}_B(len)$. The X-axis is the ratio B/len.

6.3 Security Level for MISIS in Dilithium

In the NIST post-quantum standard Dilithium, the signature forgery can be turned into an MISIS problem, which can be solved as CVP with infinity norm. Thus we can combine our discussion in Section 6.1 and Section 6.2 to estimate the hardness of MISIS in Dilithium.

We give the definition of MISIS, then describe the construction of a lattice basis which turns MISIS into approximate CVP under infinity norm.

Definition 6.1 (MISIS and ISIS). *Let $R_q = \mathbb{Z}_q[X]/(X^n+1)$ be a polynomial ring, $\mathbf{A} \leftarrow_\$ R_q^{m\times k}$ (resp. $\mathbb{Z}_q^{m\times k}$), $\mathbf{t} \leftarrow_\$ R_q^m$ (resp. $\mathbb{Z}_q^m$), the goal of solving $MISIS_{m,k,B}$ (resp. $ISIS_{m,k,B}$) is to find short vectors $\mathbf{z} \in R_q^k$ (resp. $\mathbb{Z}_q^k$), $\mathbf{u} \in R_q^m$ (resp. $\mathbb{Z}_q^m$), $\|\mathbf{z}\|_\infty \leq B_\mathbf{z}, \|\mathbf{u}\|_\infty \leq B_\mathbf{u}$, such that $\mathbf{Az} + \mathbf{u} = \mathbf{t} \mod q$.*

We note that in the document of Dilithium, Section 6.1, similar hard problems are defined with a single bound $B = \max\{B_\mathbf{u}, B_\mathbf{z}\}$, instead of different bounds for $\mathbf{u}, \mathbf{z}$. To compare with the security results in the document of Dilithium,

we first give an estimation using only the single bound B. Then we will show that considering separate bounds for $\mathbf{u}, \mathbf{z}$ can lead to a tighter estimation for the unforgeability of Dilithium.

The problem MISIS is defined on module lattices. However, there is currently no better algorithm for hard problems on module lattices compared with general lattices. To solve the MISIS problem, we should first expand $\mathbf{A}, \mathbf{t}$ into a matrix and a vector over $\mathbb{Z}_q$ through the following conversion:

For $\mathbf{A} = \begin{pmatrix} a_{0,0} & \dots & a_{0,k-1} \\ \vdots & \ddots & \vdots \\ a_{m-1,0} & \dots & a_{m-1,k-1} \end{pmatrix}$, suppose that $a_{i,j} = b_0^{(i,j)} + b_1^{(i,j)} X + \dots + b_{n-1}^{(i,j)} X^{n-1}$. We write $\mathbf{A}_{i,j} = \begin{pmatrix} b_0^{(i,j)} & -b_{n-1}^{(i,j)} & \dots & -b_1^{(i,j)} \\ b_1^{(i,j)} & b_0^{(i,j)} & \dots & -b_2^{(i,j)} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n-1}^{(i,j)} & b_{n-2}^{(i,j)} & \dots & b_0^{(i,j)} \end{pmatrix}$, which is a matrix in $\mathbb{Z}_q^{n \times n}$. Then $\mathbf{A}$ is expanded into $\begin{pmatrix} \mathbf{A}_{0,0} & \dots & \mathbf{A}_{0,k-1} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{m-1,0} & \dots & \mathbf{A}_{m-1,k-1} \end{pmatrix} \in \mathbb{Z}_q^{nm \times nk}$. For the vector $\mathbf{t}$, let $t_i = c_0^{(i)} + c_1^{(i)} X + \dots + c_{n-1}^{(i)} X^{n-1}$, $\mathbf{t}$ is expanded into $(c_0^{(0)}, \dots, c_{n-1}^{(0)}, c_0^{(1)}, \dots, c_{n-1}^{(1)}, \dots, c_0^{(m-1)}, \dots, c_{n-1}^{(m-1)})^T$ by simply putting all coefficients together.

In our discussion below, for simplicity, we let $\mathbf{A}, \mathbf{t}$ be those converted matrix and vector over $\mathbb{Z}_q$ rather than original matrix and vector over R_q. By the choice of the polynomial ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, we can see that the MISIS problem can be solved from first solving the ISIS problem for $\mathbf{A}, \mathbf{t} \in \mathbb{Z}_q^{nm \times nk} \times \mathbb{Z}_q^{nm}$ to get a solution $\mathbf{u}, \mathbf{z} \in \mathbb{Z}_q^{nm} \times \mathbb{Z}_q^{nk}$, then recovering the polynomial vectors from $\mathbf{u}, \mathbf{z}$ in a similar way.

The lattice is then constructed as follows: $\mathcal{L} = \mathcal{L}(\mathbf{B})$ where $\mathbf{B} = \begin{pmatrix} q\mathbf{I} & \mathbf{A} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$, and $\binom{\mathbf{t}}{\mathbf{0}}$ is its target vector. The lattice has its dimension $d = n(m + k)$.

We note that we may only choose a few columns in the matrix $\mathbf{A}$ to form this lattice. Although this increases the length of shortest vector of the lattice, it also decreases the dimension of the lattice, so there is a possibility that such method may accelerate the solving of the problem. However, for the parameters in Dilithium, the number of columns is rather small, and discarding columns of $\mathbf{A}$ only makes the algorithm slower.

As in the document of Dilithium [13] Section C.3, we also use a pre-processing step, which samples a random unimodular matrix $\mathbf{U}$, and use the basis $\mathbf{BU}$ instead of $\mathbf{B}$. By pre-processing, the q-vectors (those vectors with only one element q and other elements 0) do not occur in the new lattice basis $\mathbf{BU}$. This simplifies our discussion below, since it was pointed out in [15] that Geometric Series Assumption does not work well for a lattice basis with q-vectors.

We follow the document of Dilithium [13], Section 6.2.1 and Section C.3, that our goal is to find a close enough lattice vector $\mathbf{v}$ to $\binom{\mathbf{t}}{\mathbf{0}}$, such that $\binom{\mathbf{t}}{\mathbf{0}} - \mathbf{v} = \binom{-\mathbf{u}}{\mathbf{z}}$, satisfying $\|\mathbf{u}, \mathbf{z}\|_\infty \leq B$. By the construction of lattice $\mathcal{L}$, it is not hard to see that $\mathbf{u}, \mathbf{z}$ is a short solution to $\mathbf{Az} + \mathbf{u} = \mathbf{t} \mod q$.

This means that we should find a short vector $\mathbf{x}$ in $\binom{\mathbf{t}}{\mathbf{0}} + \mathcal{L}$ with the restriction $\mathcal{R}(\mathbf{x}) = 1 \leftrightarrow \|\mathbf{x}\|_\infty \leq B$, with the related probability function $\mathcal{P}_B(len)$ as defined in Section 6.2. This can be viewed as a restricted CVP instance.

The next thing is to choose the embedding parameter M in Kannan's embedding to turn this restricted CVP into restricted SVP. As discussed in Section 6.1, we can fix M to be $\sqrt{\frac{2}{d}}\ln(\sqrt{2}+1) \cdot len$ to simplify our discussion here, thus we can set

$$\mathcal{P}_{\mathrm{SVP}_\infty}(len') = \mathcal{P}_{\mathrm{CVP}_\infty}(len) \cdot (6 - 4\sqrt{2}),$$

where $len' = \sqrt{len^2 + M^2}$.

However, a new problem arises: since len' is chosen in Algorithm 8, which should be first given the description of the lattice $\mathcal{L}'$ which contains M, we cannot predetermine M or len' without having the other. We take an alternative approach:

Instead of calculating len', we calculate the fraction $len_{\mathrm{fac}} = \frac{len'}{\mathrm{vol}(\mathcal{L}')^{1/(d+1)}}$, which can be estimated solely from d, δ, κ using GSA, and is unrelated to M. Then we use len_{fac} to determine both M and len'. Since $M = \sqrt{\frac{2}{d}}\ln(\sqrt{2}+1) \cdot len$, we have that

$$len' = \sqrt{len^2 + M^2} = \sqrt{1 + (\frac{d}{2\ln^2(\sqrt{2}+1)})^2} \cdot M.$$

So given len_{fac}, we have that:

$$\sqrt{1 + \left(\frac{d}{2\ln^2(\sqrt{2}+1)}\right)^2} \cdot M = len_{\mathrm{fac}} \cdot \mathrm{vol}(\mathcal{L}')^{1/(d+1)} = len_{\mathrm{fac}} \cdot \mathrm{vol}(\mathcal{L})^{1/(d+1)} \cdot M^{1/(d+1)},$$

so

$$M^{d/(d+1)} = \frac{len_{\mathrm{fac}} \cdot \mathrm{vol}(\mathcal{L})^{1/(d+1)}}{\sqrt{1 + \left(\frac{d}{2\ln^2(\sqrt{2}+1)}\right)^2}},$$

which means that

$$M = \left(\frac{len_{\mathrm{fac}} \cdot \mathrm{vol}(\mathcal{L})^{1/(d+1)}}{\sqrt{1 + (\frac{d}{2\ln^2(\sqrt{2}+1)})^2}}\right)^{(d+1)/d}.$$

We can further calculate len' and len from M.

By determining M, we have the description of the lattice $\mathcal{L}'$ from len', which can further be used to determine the related probability $\mathcal{P}_{\mathrm{SVP}_\infty}(len')$. Then, we can run the estimation version of Algorithm 8 with proper time cost model. Here we use the gate-count model in [7], as discussed in Section 5.3.

Removing "Conservativeness" in Dilithium Forgery. The designers of Dilithium mentioned in [13], Section 6.2.1 that forging a signature is still harder than solving the MISIS problem from two aspects: first, $\mathbf{u}$ has only $\omega \in \{80, 55, 75\}$ elements larger than $\tau \cdot 2^{d-1} + \gamma_2$ (which is 254976 for Dilithium lv2), and second, $\mathbf{z}$ has its norm $\leq \gamma_1 - \beta$ (which is 130994 for Dilithium lv2). But a solution for MISIS only requires that all its elements below $\tau \cdot 2^{d-1} + 2\gamma_2 + 1$ (which is 350209 for Dilithium lv 2), which makes the problem more conservative compared with real forgery.

We can see that the first aspect does not really affect the security of Dilithium: since we solve the MISIS problem first by finding a set of short vectors of length $\leq len$, the probability that it has more than ω elements larger than $\tau \cdot 2^{d-1} + \gamma_2$ is small and can be ignored. However, the second aspect may have large impact on the security of unforgeability in Dilithium.

To remove the conservativeness from the unforgeability in Dilithium, we consider different bounds $B_{\mathbf{u}}, B_{\mathbf{z}}$ for $\mathbf{u}, \mathbf{z}$, instead of a single maximal bound. Suppose that $\mathbf{u}, \mathbf{z}$ have dimensions m, k respectively with $d = m + k$, similar to Theorem 6.2, we can see that related probability can be expressed by $\left(1 - 2\Phi\left(-\frac{\sqrt{d} \cdot B_{\mathbf{u}}}{len}\right)\right)^m \left(1 - 2\Phi\left(-\frac{\sqrt{d} \cdot B_{\mathbf{z}}}{len}\right)\right)^k$.

However, we cannot directly use this related probability to estimate the unforgeability of Dilithium. Like what have been shown in the hardness of LWE [14,22], we should use the rescaling method to balance between the lengths of $\mathbf{z}, \mathbf{u}$, in order to optimize the efficiency in solving the (M)ISIS problem.

Let δ be a rescaling factor, we can rewrite the CVP lattice with the following basis:

$$\mathbf{B}_\delta = \begin{pmatrix} \delta q \mathbf{I} & \delta \mathbf{A} \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

and $\binom{-\delta \mathbf{u}}{\mathbf{z}}$ is one of its short vectors. After rescaling, the related probability in the new lattice turns out to be $\left(1 - 2\Phi\left(-\frac{\sqrt{d} \cdot \delta B_{\mathbf{u}}}{len}\right)\right)^m \left(1 - 2\Phi\left(-\frac{\sqrt{d} \cdot B_{\mathbf{z}}}{len}\right)\right)^k$.

We give the following result for the choice of δ:

Theorem 6.3. *For solving the (M)ISIS problem through restricted SVP, the optimal rescaling factor is* $\delta = B_{\mathbf{z}}/B_{\mathbf{u}}$.

Proof. We heuristically assume that generating a random vector with length $\leq len$ has the same hardness for different lattices with the same volume. For the lattice $\mathcal{L}_\delta = \mathcal{L}(\mathbf{B}_\delta)$, let $c_\delta = (\delta q)^{-m/d}$, we have that $\mathrm{vol}(\mathcal{L}_\delta) = (\delta q)^m$, thus $\mathrm{vol}(c_\delta \mathcal{L}) = 1$, and the required vectors become $c_\delta \cdot \delta \mathbf{u}$ and $c_\delta \mathbf{z}$ with bounds $c_\delta \delta B_{\mathbf{u}}$ and $c_\delta B_{\mathbf{z}}$. Thus to optimize δ, we only need to maximize the related probability function, which is $\mathcal{P} = \left(1 - 2\Phi\left(-\frac{\sqrt{d} \cdot c_\delta \delta B_{\mathbf{u}}}{len}\right)\right)^m \left(1 - 2\Phi\left(-\frac{\sqrt{d} \cdot c_\delta B_{\mathbf{z}}}{len}\right)\right)^k$. Next, we prove that the function reaches the upper bound only when $\delta = B_{\mathbf{z}}/B_{\mathbf{u}}$.

We write the function $p(x) = \ln\left(1 - 2\Phi\left(-\frac{\sqrt{d} \cdot \exp(x)}{len}\right)\right)$, and let $a_\delta = \ln(c_\delta \delta B_{\mathbf{u}})$, $b_\delta = \ln(c_\delta B_{\mathbf{z}})$. We can further check that $\frac{m}{d} a_\delta + \frac{k}{d} b_\delta = \frac{m}{d}(\ln B_{\mathbf{u}} - \ln q) + \frac{k}{d} \ln B_{\mathbf{z}} = a_{B_{\mathbf{z}}/B_{\mathbf{u}}} = b_{B_{\mathbf{z}}/B_{\mathbf{u}}}$ is unrelated with δ.

It seems that there is no analytical expression for $p(x)$. To show that $p(x)$ is a concave function, we draw the graph of $p''(x)$ with the help of Wolfram Alpha (in fact, we draw the graph of $p''(x+C)$ where $C = \ln\left(\frac{\sqrt{2d}}{len}\right)$ instead, so that the function does not contain undetermined variables):

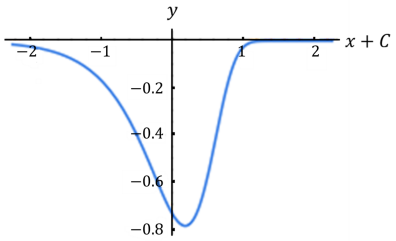

Fig. 5. Function graph of $p''(x+C)$.

From Figure 5, we have confidence that $p''(x) < 0$, so by the property of concave functions, $\frac{m}{d}p(a_\delta)+\frac{k}{d}p(b_\delta) \leq p\left(\frac{m}{d}a_\delta + \frac{k}{d}b_\delta\right) = p\left(\frac{m}{d}(\ln B_{\mathbf{u}} - \ln q) + \frac{k}{d}\ln B_{\mathbf{z}}\right)$, where the equality holds only when $\delta = B_{\mathbf{z}}/B_{\mathbf{u}}$.

Since the related probability $\mathcal{P} = \exp\left(d\left(\frac{m}{d}p(a_\delta) + \frac{k}{d}p(b_\delta)\right)\right)$, we have that $\mathcal{P}$ also takes its maximal value only when $\delta = B_{\mathbf{z}}/B_{\mathbf{u}}$. Thus we finish the proof. □

We note that rescaling should not be applied to the lattice $\mathcal{L}'$ after Kannan's embedding.

We reestimate the hardness of Dilithium after rescaling to give a non-conservative estimation, and find out that the result is much higher than the conservative estimation. We summarize our estimation results in Table 1. The code of our estimator can be found in https://github.com/Summwer/inf-cvp-estimator.

Table 1. Estimation Result for MISIS in Dilithium

	Estimation in [13]		Our Estimation (conservative)				Our Estimation (non-conservative)			
	β	bits	β	κ/f'	$\log(S)$	bits	β	κ/f'	$\log(S)$	bits
Lv2	423	123	436	447/11	-	159.9	478	485/10	-	171.5
Lv3	638	186	661	-	150.2	223.0	677	-	159.5	227.7
Lv5	909	265	936	-	229.6	300.3	963	-	236.9	307.9

Here, the conservative estimation means the hardness of solving the MISIS problem $\mathbf{Az} + \mathbf{u} = \mathbf{t} \mod q$ with $\|\mathbf{z}, \mathbf{u}\|_\infty \leq \max\{B_{\mathbf{u}}, B_{\mathbf{z}}\}$ as in the document of Dilithium, which is insufficient for a signature forgery. The non-conservative estimation means the hardness of solving the same MISIS problem with $\|\mathbf{u}\|_\infty \leq B_{\mathbf{u}}, \|\mathbf{z}\|_\infty \leq B_{\mathbf{z}}$ (using the rescaling factor $B_{\mathbf{z}}/B_{\mathbf{u}}$) that can be turned into a

signature forgery. Compared with the conservative estimation, we can use the non-conservative estimation to get a tighter security result for Dilithium.

From Table 1 above, we can see that our first estimation fits well with the estimation in the document of Dilithium [13], considering the differences between gate-count and core-SVP model. However, our second estimation which makes use of rescaling is much higher than the first one, so we claim that using one maximal bound instead of different bounds for $\mathbf{u}, \mathbf{z}$ does bring much conservativeness in Dilithium.

In Dilithium lv2, the optimal choice of solving MISIS is through the flexible-d4f approach, while in Dilithium lv3 and lv5, the optimal choice is through the sieve-then-slice approach.

We note that we do not specify the value of κ in Dilithium lv3 and lv5, since the optimal value of κ is in fact smaller than β, and different values of κ lead to very close estimation results as long as $\kappa << \beta$. The reason is that under the parameters of Dilithium v3 and v5, solving SVP from two-step mode is less efficient than using only BKZ, so the time cost is in fact dominated by the cost of BKZ, while the κ-dimensional sieve only has its effect in generating enough lattice vectors. Thus changing the value of κ only affects the estimation result by little.

Discussion: Enhancing the Unforgeability of Dilithium. In our algorithm, either from the flexible-d4f or the sieve-then-slice approach, we both need to find a set of relatively short lattice vectors with lengths $\leq len$, then find a vector among them within the bound B. However, $\mathrm{vol}(\mathcal{B}^d(len))$ is in fact much smaller than $\mathrm{vol}([-B:B]^d)$, so it should be easy for a dectector to detect whether a signature is forged from these approaches. So if we add one more rejection condition into Dilithium, that the signature is rejected if $\|\mathbf{u}, \mathbf{z}\| < len$ for a small len, we can totally ban both approaches for forging the signature, while the additional rejection probability is small. It might still be possible that we use Gaussian sampling (discussed in Section 5.2) to solve the problem, but it should be much slower than the two approaches used in our restricted SVP solver, which means that the signature may have better unforgeability.

7 Conclusion and Future Work

In this paper, we define a new lattice hard problem, called restricted SVP, which can be viewed as a variant of approximate SVP, and present an heuristic solver for this problem with two novel techniques, called flexible dimension for free and sieve-then-slice. We show that restricted SVP can be a generalization for may types of lattice problems, including Kannan's embedding on approximate CVP and SVP with infinity norm, by formally define their restriction predicate as well as the probability for a lattice vector to pass the restrictions. Then we combine the two problems together to get an estimator for the security of signature unforgeability in Dilithium, as signature forgery in Dilithium can be turned into the MISIS problem, which can be solved by CVP with infinity norm.

We show that our estimator can give a more accurate estimation for Dilithium under gate-count model.

There are still many problems left to be solved, we list a few of them: (1) Since we only present an heuristic algorithm, is there a more efficient algorithm for solving restricted SVP, or can we give a strict lower bound for the hardness of this algorithm? (2) Are there more lattice problems which can be turned into restricted SVP to get a more efficient solving algorithm? (3) Can we present an algorithm for solving the restricted uSVP (or LWE) problem? We leave these problems as our future work.

Acknowledgments

The authors thank the reviewers of PQCrypto 2025 for their helpful comments to improve the contents of this paper. This work is supported by the National Natural Science Foundation of China No. U2336210 and Ant Group.

References

1. Divesh Aggarwal and Priyanka Mukhopadhyay. Improved algorithms for the shortest vector problem and the closest vector problem in the infinity norm. In Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao, editors, *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*, volume 123 of *LIPIcs*, pages 35:1–35:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
2. Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 284–293. ACM, 1997.
3. Martin R. Albrecht, Shi Bai, Pierre-Alain Fouque, Paul Kirchner, Damien Stehlé, and Weiqiang Wen. Faster enumeration-based lattice reduction: Root hermite factor $k^{1/(2k)}$ time $k^{k/8+o(k)}$. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 186–212. Springer, 2020.
4. Martin R. Albrecht, Shi Bai, Jianwei Li, and Joe Rowell. Lattice reduction with approximate enumeration oracles - practical algorithms and concrete performance. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 732–759. Springer, 2021.
5. Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The general sieve kernel and new records in lattice reduction. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 717–746. Springer, 2019.

6. Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-svp. In Hyang-Sook Lee and Dong-Guk Han, editors, *Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers*, volume 8565 of *Lecture Notes in Computer Science*, pages 293–310. Springer, 2013.
7. Martin R. Albrecht, Vlad Gheorghiu, Eamonn W. Postlethwaite, and John M. Schanck. Estimating quantum speedups for lattice sieves. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 583–613. Springer, 2020.
8. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 327–343. USENIX Association, 2016.
9. Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 57–74. Springer, 2013.
10. Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 789–819. Springer, 2016.
11. Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber algorithm specifications and supporting documentation (version 3.0). *NIST PQC Round 3 submissions*, 2020.
12. L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, March 1986.
13. Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehl'e. Crystals-dilithium algorithm specifications and supporting documentation. *NIST PQC Round 3 submissions*, 2020.
14. Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *Information Security and Privacy - 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings*, volume 8544 of *Lecture Notes in Computer Science*, pages 322–337. Springer, 2014.
15. Shi Bai, Damien Stehlé, and Weiqiang Wen. Measuring, simulating and exploiting the head concavity phenomenon in BKZ. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 369–404. Springer, 2018.

16. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 10–24. SIAM, 2016.
17. Anja Becker, Nicolas Gama, and Antoine Joux. Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search. *IACR Cryptol. ePrint Arch.*, page 522, 2015.
18. Huck Bennett, Mahdi Cheraghchi, Venkatesan Guruswami, and João Ribeiro. Parameterized inapproximability of the minimum distance problem over all fields and the shortest vector problem in all l_p norms. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 553–566. ACM, 2023.
19. Hao Chen. A measure version of gaussian heuristic. *IACR Cryptol. ePrint Arch.*, page 439, 2016.
20. Yuanmi Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD Thesis, 2013.
21. Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.
22. Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 329–358. Springer, 2020.
23. Emmanouil Doulgerakis, Thijs Laarhoven, and Benne de Weger. Finding closest lattice vectors using approximate voronoi cells. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*, volume 11505 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2019.
24. Emmanouil Doulgerakis, Thijs Laarhoven, and Benne de Weger. Sieve, enumerate, slice, and lift: - hybrid lattice algorithms for SVP via CVPP. In Abderrahmane Nitaj and Amr M. Youssef, editors, *Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings*, volume 12174 of *Lecture Notes in Computer Science*, pages 301–320. Springer, 2020.
25. Léo Ducas. Shortest vector from lattice sieving: A few dimensions for free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 125–145. Springer, 2018.
26. Léo Ducas, Thijs Laarhoven, and Wessel P. J. van Woerden. The randomized slicer for CVPP: sharper, faster, smaller, batchier. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of*

Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II, volume 12111 of *Lecture Notes in Computer Science*, pages 3–36. Springer, 2020.
27. Léo Ducas, Marc Stevens, and Wessel P. J. van Woerden. Advanced lattice sieving on gpus, with tensor cores. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 249–279. Springer, 2021.
28. Thomas Espitau and Paul Kirchner. The nearest-colattice algorithm: Time-approximation tradeoff for approx-CVP. *Open Book Series*, 4(1):251–266, December 2020.
29. Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.
30. Elena Kirshanova and Thijs Laarhoven. Lower bounds on lattice sieving and information set decoding. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 791–820. Springer, 2021.
31. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.
32. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982.
33. MATZOV. Report on the Security of LWE: Improved Dual Lattice Attack. https://doi.org/10.5281/zenodo.6412487, April 2022.
34. Priyanka Mukhopadhyay. Faster provable sieving algorithms for the shortest vector problem and the closest vector problem on lattices in lp norm. *Algorithms*, 14(12):362, 2021.
35. Phong Q. Nguyen. Hermite's constant and lattice algorithms. In Phong Q. Nguyen and Brigitte Vallée, editors, *The LLL Algorithm - Survey and Applications*, Information Security and Cryptography, pages 19–69. Springer, 2010.
36. Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *J. Math. Cryptol.*, 2(2):181–207, 2008.
37. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.
38. Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In Lothar Budach, editor, *Fundamentals of Computation Theory, 8th International Symposium, FCT '91, Gosen, Germany, September 9-13, 1991, Proceedings*, volume 529 of *Lecture Notes in Computer Science*, pages 68–85. Springer, 1991.
39. Wenwen Xia, Leizhang Wang, Geng Wang, Dawu Gu, and Baocang Wang. Improved progressive BKZ with lattice sieving. *IACR Cryptol. ePrint Arch.*, page 1343, 2022.
40. Wenwen Xia, Leizhang Wang, Geng Wang, Dawu Gu, and Baocang Wang. A refined hardness estimation of LWE in two-step mode. In Qiang Tang and Vanessa Teague, editors, *Public-Key Cryptography - PKC 2024 - 27th IACR International Conference on Practice and Theory of Public-Key Cryptography, Sydney, NSW, Australia, April 15-17, 2024, Proceedings, Part III*, volume 14603 of *Lecture Notes in Computer Science*, pages 3–35. Springer, 2024.