On the Asymptotic Complexity of Solving LWE

Gottfried Herold, Elena Kirshanova, and Alexander May

Horst Görtz Institute for IT-Security
Faculty of Mathematics
Ruhr University Bochum, Germany
elena.kirshanova@rub.de

Abstract. We provide for the first time an asymptotic comparison of all known algorithms for the search version of the Learning with Errors (LWE) problem. This includes an analysis of several lattice-based approaches as well as the combinatorial BKW algorithm. Our analysis of the lattice-based approaches defines a general framework, in which the algorithms of Babai, Lindner-Peikert and several pruning strategies appear as special cases. We show that within this framework, *all* lattice algorithms achieve the same asymptotic complexity.
For the BKW algorithm, we present a refined analysis for the case of only a polynomial number of samples via amplification, which allows for a fair comparison with lattice-based approaches. Somewhat surprisingly, such a small number of samples does not make the asymptotic complexity significantly inferior, but only affects the constant in the exponent.
As the main result we obtain that both, lattice-based techniques and BKW with a polynomial number of samples, achieve running time $2^{\mathcal{O}(n)}$ for n-dimensional LWE, where we make the constant hidden in the big-$\mathcal{O}$ notion explicit as a simple and easy to handle function of all LWE-parameters. In the lattice case this function also depends on the time to compute a BKZ lattice basis with block size $\Theta(n)$. Thus, from a theoretical perspective our analysis reveals how LWE's complexity changes as a function of the LWE-parameters, and from a practical perspective our analysis is a useful tool to choose LWE-parameters resistant to all known attacks.

Keywords. LWE security, Bounded Distance Decoding, Lattices, BKW

1 Introduction

Lattice-based cryptosystems are currently the best candidates for post-quantum security and allow for a variety of new functionalities. This led to an impressive amount of publications within the last decade (see e.g. [43],[19], [41] and their follow-up works). The security of most lattice-based public-key encryption schemes relies on the hardness of the *Learning with Errors* (LWE) problem – an average-case hard lattice problem introduced into cryptography by Regev ([43]), and formerly studied in the learning community ([21, 13]). In the search version of LWE, one has to recover a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ by looking at m samples $(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)$, where $\mathbf{a}_i \in \mathbb{Z}_q^n$ is chosen uniformly at random and e_i is a discrete Gaussian error with (scaled) standard deviation s. Notice that the input size of such an LWE sample is linear in $n, \log q$ and $\log s$.

LWE has been proven to be as hard as quantumly approximating worst-case instances of the so-called *Shortest Independent Vectors Problem* (SIVP) up to polynomial factors ([43]). SIVP with polynomial approximation factors is not NP-hard under

standard complexity assumptions ([20]), but the currently best algorithms for finding shortest vectors even with polynomial approximation ratio require either exponential time and space $2^{\mathcal{O}(n)}$ ([3, 39]) or for polynomial space slightly super-exponential time $2^{\mathcal{O}(n\log n)}$ ([26]).

While Regev's quantum reduction from SIVP to LWE is dimension-preserving, Peikert's classical reduction [40] has a quadratic loss by transforming n-dimensional lattice problems into n^2-dimensional LWE instances in *polynomial* time (see also [14]) . These complexity theoretic results stress the need to study LWE's complexity directly in order to be able to instantiate LWE in cryptography with a concrete predetermined security level.

For the LPN problem, which is a special case of LWE for $q = 2$, Blum, Kalai and Wasserman (BKW, [13]) designed the currently fastest algorithm with slightly sub-exponential complexity $2^{\mathcal{O}(\frac{n}{\log n})}$(see also the discussion in Regev [43]). Unfortunately, BKW requires the same sub-exponential amount of memory *and* of LPN samples. Recent research [4, 5] analyzes BKW's complexity in the LWE-setting for $q = \text{poly}(n)$, where the authors provide fully exponential (in n) bounds for the runtime, as well as for memory and sample complexity. BKW's huge sample complexity makes the algorithm often useless for LWE cryptanalysis in practice, since cryptographic primitives like encryption usually provide no more than a polynomial number of samples to an attacker.

Concerning lattice-based attacks, the algorithm of Lindner and Peikert [32], which is an adaptation of Babai's `NearestPlane` algorithm to the LWE setting, and its improvements due to Liu-Nguyen ([33]) and Aono et al. ([8]) are considered as the practical benchmark standard in cryptanalysis to assess LWE's security. Unfortunately, the authors do not provide an asymptotic analysis of their algorithms as a function of the LWE parameters (n, m, q, s). This is an unsatisfactory situation, since it makes lattice-based approaches somewhat incomparable among themselves and especially to the combinatorial BKW-algorithm in the LWE scenario [4, 7, 29]. The lattice-based literature often suggests that lattice-based approaches are most practical for attacking LWE, while the BKW literature suggests that asymptotically the BKW algorithm always outperforms lattice reduction. Our results show that both statements should be taken with care. It really depends on the LWE-parameters (and the lattice reduction algorithm) which approach is asymptotically fastest, even when we use BKW restricted to only a linear number m of samples.

Whether LWE-type cryptosystems will eventually be used in practice crucially depends on a good understanding of the complexity of LWE-instances. A proper cryptanalytic treatment of a complexity assumption such as LWE includes practical cryptanalysis of reasonably sized instances as well as an extrapolation to cryptographic security level instances from *asymptotic formulas*. This is the widely accepted approach for estimating key sizes [30, 1], which is for instance taken for measuring the hardness of factoring RSA-1024 [28]. Whereas some practical experiments on concrete LWE instances where reported in the literature [32, 33, 8], the asymptotics remains unclear. Our work fills this gap.

Outline of our results. In a survey-paper [6], Albrecht et al. conclude that 'for most algorithms, there is no sufficiently precise closed formula which expresses the running time in terms of the parameters'. We clarify this issue by presenting a unified com-

plexity analysis of the LWE problem that covers techniques such as lattice-based approaches (lattice-basis reduction+enumeration, embedding) and the combinatorial BKW-approach. We state the algorithmic complexity regarding the three metrics: expected running time, memory complexity and number of LWE samples, all as a function of the LWE-parameter (n,q,s). For attaining our results, we introduce the following techniques.

1. We propose a new generalized framework for Bounded Distance Decoding (BDD) enumeration techniques, which we call *generalized pruning*. Our framework covers all previously known enumeration approaches such as Babai's NearestPlane algorithm, its LWE-extension NearestPlanes of Lindner-Peikert ([32]), Linear Pruning ([45]) and Extreme Pruning ([17]), which were left without rigorous analysis in [6]. We show that *all* these approaches achieve *the same* expected running-time $\Upsilon = 2^{cf(n)}$ having *the same* constant c. Of course, we should stress once again that our analysis is purely asymptotic (as opposed to [6]). So in practice some pruning strategies clearly outperform others, as reported in several experiments in the literature [32, 33, 8], but our analysis shows that this superiority vanishes for increasing n.
   To provide a complete picture of lattice-based attacks, in Sect. 5 we also include the asymptotic complexity analysis of LWE using Kannan's embedding technique [26]. See Table 1 for the precise complexity estimates.
2. In Sect. 7 we refine the asymptotic complexity analysis of the BKW algorithm in the case where only $m = \mathcal{O}(n \log n)$ samples are given. This amplification of samples is similar to Lyubashevsky's amplification for the LPN-case [34]. However, whereas the LPN-case amplification raised the running time from $2^{\mathcal{O}(\frac{n}{\log n})}$ to $2^{\mathcal{O}(\frac{n}{\log\log n})}$, in the LWE case the loss affects only the constant. Again, see Table 1 for a more precise statement of the running time as a function of the LWE-parameters.

Since any lattice-based attack relies on a basis-reduction as preprocessing, its running time crucially depends on the complexity of basis reduction. The basis reduction algorithms of Kannan ([25]) or MV, ADRS, Laarhoven ([39, 2, 29]) have run time complexities $2^{\Theta(n \log n)}$ or $2^{\Theta(n)}$, respectively. We denote in Table 1 by c_{BKZ} the hidden constant in these run time complexities. We write the LWE parameters $q = n^{c_q}$, $s = n^{c_s}$, where typically c_q, c_s are constants in cryptographic settings. For completeness, we also include the performance of the Arora-Ge algorithm ([10]) in Table 1, an algebraic attack on LWE that achieves sub-exponential running time for $c_s < \frac{1}{2}$.

Notice that single exponential lattice reduction algorithms such as MV and ADRS ([39, 2]) lead to running time 2^{cn} as in the BKW-case. Also from Table 1 one can already see that lattice-based attacks asymptotically outperform BKW (even with an exponential number of samples) as soon as the lattice reduction exponent c_{BKZ} is small enough, since in all lattice attacks the denominator is quadratic in $c_q - c_s$, whereas it is linear in $c_q - c_s$ for BKW. Our results show quantitatively how the hunt for the best run time exponent for lattice reduction [39, 2, 29] directly affects LWE's security.

In Figure 1 we compare the behaviour of the constant c for various algorithms and typical values. Here, the (heuristic) probabilistic sieving method of Laarhoven ([29]) with a runtime exponent of $c_{\mathrm{BKZ}} = 0.33$ already outperforms the BKW algorithm for some parameter-sets.

Table 1: Asymptotic comparison of LWE decoding algorithms. We denote $q = \mathcal{O}(n^{c_q})$, $s = \mathcal{O}(n^{c_s})$ and $c_{\mathtt{BKZ}}$ is the constant hidden in the run time exponent of lattice reduction. For Arora-Ge, $2 \leq \omega < 3$ is the linear algebra constant.

	Complexity				
	$\Upsilon = \frac{T(\mathrm{ALG})}{P_{\mathrm{succ}}(\mathrm{ALG})}$, M = Space				#Samples
	$T_{\mathtt{BKZ}} = 2^{c_{\mathtt{BKZ}} n \log n}$, $\mathsf{M}_{\mathtt{BKZ}} = \mathrm{poly}(n)$		$T_{\mathtt{BKZ}} = 2^{c_{\mathtt{BKZ}} n}$, $\mathsf{M}_{\mathtt{BKZ}} = 2^{\Theta(n)}$		
	$\log(\Upsilon)$	M	$\log(\Upsilon)$	M	
ENUM: – Babai (Sect. 4.1) – Lindner-P. (Sect. 4.2) – `GenPruning` (S. 4.3)	$\frac{2c_{\mathtt{BKZ}} \cdot c_q}{(\sqrt{2c_{\mathtt{BKZ}}} + c_q - c_s)^2} n \log n$	$\mathrm{poly}(n)$	$\frac{2c_{\mathtt{BKZ}} \cdot c_q}{(c_q - c_s)^2} n$	$2^{\Theta(n)}$	$\Theta(n)$
DUAL (Sect. 6)	$\frac{2c_{\mathtt{BKZ}} \cdot c_q}{(c_q - c_s)^2} \cdot n \log n$ $\frac{2c_{\mathtt{BKZ}} \cdot c_q}{(c_q - c_s + 1/2)^2} \cdot n \log n$	$\mathrm{poly}(n)$	$\frac{2c_{\mathtt{BKZ}} \cdot c_q}{(c_q - c_s)^2} \cdot n$ $\frac{2c_{\mathtt{BKZ}} \cdot c_q}{(c_q - c_s + 1/2)^2} \cdot n$	$2^{\Theta(n)}$	$\Theta(n \log n)$ $2^{\Theta(n)}$
Embedding (Sect. 5)	$\frac{2c_{\mathtt{BKZ}} c_q}{(c_q - c_s)^2} \cdot n \log n$	$\mathrm{poly}(n)$	$\frac{2c_{\mathtt{BKZ}} c_q}{(c_q - c_s)^2} \cdot n$	$2^{\Theta(n)}$	$\Theta(n)$

	$\log(\Upsilon)$	M	#Samples
`BKW` ([4])	$\frac{1}{2} \frac{c_q}{c_q - c_s + 1/2} \cdot n$	$2^{\Theta(n)}$	$2^{\Theta(n)}$
`BKW` (Sect. 7)	$\frac{1}{2} \frac{c_q}{c_q - c_s} \cdot n$	$2^{\Theta(n)}$	$\Theta(n \log n)$
`BKW2` ([27, 22])	$\left(1/c_q + 2\ln(c_q/c_s)\right)^{-1} \cdot n$	$2^{\Theta(n)}$	$2^{\Theta(n)}$
`BKW2` (Sect. 7)	$\left(2\ln(c_q/c_s)\right)^{-1} \cdot n$	$2^{\Theta(n)}$	$\Theta(n \log n)$
Arora-Ge ([6]), $c_s < 1/2$	$\omega \cdot (1 - 2c_s) \cdot n^{2c_s} \log^2(n)$	$\mathcal{O}(2^{n^{c_s} \log^2 n})$	
Arora-Ge ([6]), $c_s \geq 1/2$	$\omega \cdot (2c_s - 1) \cdot n \log(n)$	$\mathcal{O}(2^{n \log n})$	

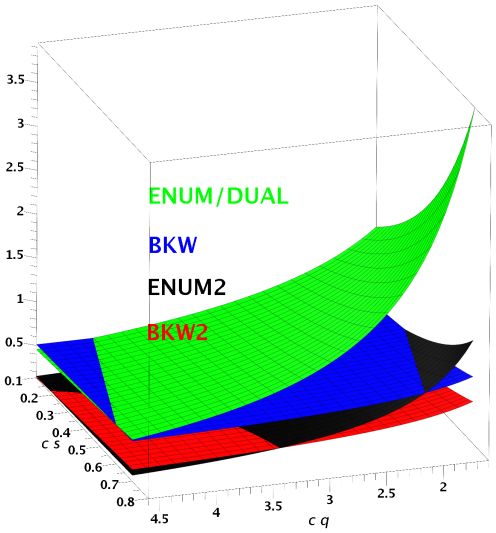


Fig. 1: Running time exponents for single-exponential attacks on LWE with *polynomial* number of samples for parameters $c_q \in [1.6 \ldots 4.6]$ and $c_s \in [0.1 \ldots 0.7]$. For reduction step we have $c_{\mathtt{BKZ}} = 1$ [2] (green plot) and allowing heuristics as in [29] $c_{\mathtt{BKZ}} = 0.33$ (black plot). The `BKW` algorithm from [4] is in blue, the red plot is a recent algorithm of [27], [22].

2 Background

We use bold lower-case letters for vectors $\mathbf{b}$ and we let $\|\mathbf{b}\|$ denote their Euclidean norm. We compose vectors column-wise into matrices. For a linearly independent set $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{R}^n$, the *fundamental domain* $\mathscr{P}_{1/2}(\mathbf{B})$ is $\left\{\sum_{i=1}^k c_i \mathbf{b}_i \colon c_i \in [-\frac{1}{2}, \frac{1}{2})\right\}$. The *Gram-Schmidt orthogonalisation (basis)* $\widetilde{\mathbf{B}} = (\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_k)$ is obtained iteratively by setting $\widetilde{\mathbf{b}}_1 = \mathbf{b}_1$ and $\widetilde{\mathbf{b}}_i$ as the orthogonal projection of $\mathbf{b}_i$ on $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^\perp$ for $i = 2, \ldots, k$. In this work we deal with so-called q-ary lattices Λ (i.e. $q\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$) generated by a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{Z}_q^n$:

$$\Lambda = \mathscr{L}(\mathbf{B}) = \left\{ \sum_{i=1}^n z_i \cdot \mathbf{b}_i \mod q \colon z_i \in \mathbb{Z} \right\}.$$

There are several hard problems that we can instantiate on lattices. The closest vector problem (CVP) asks to find a lattice point $\mathbf{v}$ closest to a given point $\mathbf{t} \in \mathbb{R}^n$. In the promise variant of this problem, known as *Bounded Distance Decoding* (BDD), we have some bound R on the distance between the lattice and $\mathbf{t}$: $\|\mathbf{v} - \mathbf{t}\| \leq R$, where R is usually much smaller than the lattice's packing radius.

For two discrete random variables X and Y with range S, the *statistical distance* between X and Y is $\mathrm{SD}(X;Y) = \frac{1}{2}\sum_{s \in S} |\Pr[X = s] - \Pr[Y = s]|$. The min-entropy function is denoted $H_\infty(X) = -\log \max_{s \in S}\{\Pr[X = s]\}$.

Discrete Gaussian Distribution. To each lattice-vector $\mathbf{v} \in \Lambda$ we can assign a probability proportional to $\exp(-\pi\|\mathbf{v}\|^2/s^2)$ for the Gaussian parameter[1] $s > 0$ (see e.g. [19] for a sampling algorithm). We call the resulting distribution having Λ as support the *discrete Gaussian* distribution.

For integer lattices, a sufficiently wide discrete Gaussian blurs the discrete structure of Λ (more formally, $s = \mathrm{poly}(n)$ exceeds the smoothing parameter [37] of $\mathbb{Z}^n$ for $c_s > \frac{1}{2}$), such that the distribution becomes very close to a continuous Gaussian ([41], [32]). In our analysis, we make use of continuous Gaussians to estimate the success probability of our decoding algorithms.

We use the following well-known tail bound for Gaussians. For fixed s and $y \to \infty$:

$$1 - \frac{\int_{-y}^{y} \exp(-\frac{\pi x^2}{s^2})\mathrm{d}x}{s} = e^{-\Theta(\frac{y^2}{s^2})} \qquad 1 - \frac{\sum_{x=-y}^{y} \exp(-\frac{\pi x^2}{s^2})}{\sum_{x=-\infty}^{\infty} \exp(-\frac{\pi x^2}{s^2})} = e^{-\Theta(\frac{y^2}{s^2})} \quad . \tag{1}$$

Learning with Errors. The *Learning with Errors* problem ([43]) is parametrized by a dimension $n \geq 1$, an integer modulus $q = \mathrm{poly}(n)$ and an error distribution D on $\mathbb{Z}$. Typically, D is a discrete Gaussian on with parameter s. For secret $\mathbf{s} \in \mathbb{Z}_q^n$, an LWE sample is obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, an error $e \leftarrow D$, and outputting the pair $(\mathbf{a}, t = \langle \mathbf{a}, \mathbf{s} \rangle + e \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. Having a certain number m of such pairs, we can write this problem in matrix form as $(\mathbf{A}, \mathbf{t} = \mathbf{A}^\mathsf{t}\mathbf{s} + \mathbf{e} \bmod q)$ for $\mathbf{t} = (t_1, \ldots, t_m)$, $\mathbf{e} = (e_1, \ldots, e_m)$ and the columns of matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ are composed of the $\mathbf{a}_i$. Overall, the LWE problem is given by parameters (n, q, s) and m (this parameter we can choose ourselves). Typically, (n, q, s) are related as $q = \mathscr{O}(n^{c_q}), s = \mathscr{O}(n^{c_s})$,

[1] For $s \to \infty$, the standard deviation is $s/\sqrt{2\pi} + o(s)$, the $o(s)$ being due to discretization.

and $0 < c_s < c_q$ are constants. If not specified otherwise, we assume these relations throughout.

The *search* version of the LWE problem asks to find $\mathbf{s}$, the *decision* version asks to distinguish $\mathbf{t}$ from a uniform vector, given $\mathbf{A}$. The LWE problem is an average-case hard Bounded Distance Decoding problem for the q-ary lattice $\Lambda(\mathbf{A}^t) = \{\mathbf{z} \in \mathbb{Z}^m : \exists \mathbf{s} \in \mathbb{Z}_q^n$ s.t. $\mathbf{z} = \mathbf{A}^t\mathbf{s} \bmod q\}$. Assuming $\mathbf{A}$ is full-rank, its determinant is $\det(\Lambda(\mathbf{A}^t)) = q^{m-n}$.

Lattice basis reduction. The goal of *lattice basis reduction* algorithms is to make some input basis as short and orthogonal as possible. The algorithm that is most relevant in practice is a block-wise generalization of the LLL-algorithm, called BKZ algorithm. There are two approaches to express the complexity of the BKZ algorithm. The first approach is via the so-called *Hermite factor* defined as $\delta^m = \|\mathbf{b}_1\| / \operatorname{vol}(L)^{\frac{1}{m}}$, for an m-dimensional lattice L, where $\mathbf{b}_1$ is the shortest vector of the output basis. The Hermite factor, introduced in [18], indicates how orthogonal the output basis is (we have $\delta \geq 1$, with equality for an orthogonal basis).

The second approach – rather than relying on the output parameter δ – relates the running time of the BKZ to the input block-size β. As a subroutine, BKZ calls an SVP-solver in a sub-lattice of dimension β. In [23], the authors show that after a polynomial (in m) number of SVP-calls, BKZ will produce a basis where the first (i.e. shortest) vector satisfies

$$\|\mathbf{b}_1\| \leq 2(\beta)^{\frac{m}{2\beta}} \cdot (\det L)^{\frac{1}{m}}. \tag{2}$$

Thus, the running time of BKZ is $T_{\mathsf{BKZ}} = \operatorname{poly}(m) \cdot T_{\mathsf{SVP}}(\beta)$, where $T_{\mathsf{SVP}}(\beta)$ is the running time of an SVP-solver in dimension β. With current algorithms, it is at least exponential in β, and has been improved from $2^{O(\beta^2)}$ ([16]), to $2^{O(\beta \log \beta)}$ in [25], and recently to $2^{O(\beta)}$ in [39] (the latter has also $2^{O(\beta)}$ memory complexity). There is no analogous result proven for the complexity of BKZ in terms of δ.

Geometric Series Assumption (GSA), proposed by Schnorr ([44]), provides an estimate on the length of the Gram-Schmidt vectors of a BKZ-reduced basis $\mathbf{B}$. It assumes that the sequence of $\|\widetilde{\mathbf{b}}_i\|$'s decays geometrically in i, namely, $\frac{\|\widetilde{\mathbf{b}}_i\|}{\|\widetilde{\mathbf{b}}_{i+1}\|} \approx \delta^2$. Thus, GSA allows us to predict the lengths of all Gram-Schmidt vectors as $\|\widetilde{\mathbf{b}}_i\| \approx \|\mathbf{b}_1\| \cdot \delta^{2(1-i)}$. From the analysis of [23], it follows that in terms of β, GSA can be stated as

$$\|\widetilde{\mathbf{b}}_i\| \approx \|\mathbf{b}_1\| \cdot \beta^{-\frac{i}{\beta}}. \tag{3}$$

In our asymptotic analysis we treat the above Eq. (3) as an *equality*[2], which is the worst-case for the length of the shortest vector returned by BKZ reduction: we have $\|\mathbf{b}_1\| = \beta^{\frac{m}{2\beta}} \cdot (\det L)^{\frac{1}{m}}$. Equivalently, for an LWE lattice $\Lambda(\mathbf{A}^t)$, $\|\mathbf{b}_1\| = \beta^{\frac{m}{2\beta}} \cdot q^{1-\frac{n}{m}}$. This follows from the fact that the product of all Gram-Schmidt vectors is equal to the lattice-determinant.

We note here that according to [23], the above relation should only hold for the first $m - \beta$ Gram-Schmidt vectors. Indeed, a worst-case analysis [24] shows that the last Gram-Schmidt vectors behave like $\|\widetilde{\mathbf{b}}_i\| \approx \exp(-\frac{1}{4}\log(d-i)^2)$, showing a faster decay than GSA suggests. In this paper, however, we stick to Eq. (3), as it greatly simplifies the exposition. Note that we can ameliorate the effect of this discrepancy on our analysis

[2] Our runtime analysis is in all cases robust against small deviations from exact equality.

by BKZ-reducing the dual of the lattice and taking the dual of the returned basis, so the faster decay occurs during the first (rather than the last) Gram-Schmidt vectors.

3 LWE Decoding

This section gives an extended roadmap to the subsequent sections: we briefly describe the existing methods to solve the *search* LWE problem, for each of which we present a complexity analysis. For all the approaches considered, we are interested in the quantity $\Upsilon(\mathrm{ALG}) = \frac{T(\mathrm{ALG})}{P_{\mathrm{succ}}(\mathrm{ALG})}$, the time/success trade-off. The decoding of an LWE instance $(\mathbf{A}, \mathbf{t} = \mathbf{A}^{\mathbf{t}}\mathbf{s} + \mathbf{e} \bmod q)$ is successful, when the returned error-vector is indeed $\mathbf{e}$ (or, equivalently, we return the lattice-vector $\mathbf{A}^{\mathbf{t}}\mathbf{s}$).

Currently, there are three conceptually different ways to approach LWE: lattice-based methods, combinatorial methods and algebraic methods. The lattice-based methods, in turn, can also be divided into three approaches: we can first view LWE as a BDD instance for the lattice $\Lambda(\mathbf{A}^{\mathbf{t}})$, or second we apply Kannan's homogenization technique to convert the BDD instance into a unique-SVP instance by adding the target vector $\mathbf{t}$ to a basis of $\Lambda(\mathbf{A}^{\mathbf{t}})$ and search for the shortest vector in a lattice of increased dimension, or third, we can target the decision LWE problem by solving approximate-SVP in the dual lattice. In this work, we primarily focus on BDD, while presenting only shortly the complexity of unique-SVP embedding for LWE in Sect. 5 and the dual approach in Sect. 6.

As for combinatorial BKW-type methods ([13], [4]), to recover $\mathbf{s}$ we apply a Gaussian elimination approach in the presence of errors, where we allow to multiply our equations only by ± 1 to keep the error small. In other words, we query LWE samples $(\mathbf{a}, t)$ until we find pairs that match (up to sign) on some fixed coordinates, add them up to obtain zeros on these coordinates and proceed on the remaining non-zero parts. Once a sample with only one non-zero coordinate $(a'_0 s_0 + e'_0, t_0)$ is obtained, we brute-force on s_0 (the error e'_0, being the sum of all the errors used to generate this sample, is very large compared to the initial e_i's and we cannot conclude on s_0 immediately). Recent improvements ([27, 22]) only require small coordinates (rather than zero). As opposed to the lattice-based attacks, these BKW-type methods succeed with high probability on a random matrix $\mathbf{A}$, provided we can query for exponentially many samples. This condition, however, is unrealistic: the number of LWE samples exposed by a primitive is typically only $\mathrm{poly}(n)$, possibly only $\mathcal{O}(n)$. Thus, in Sect. 7, we are mainly concerned with the analysis of the so-called *amplification* technique, where out of $\Theta(n \log n)$ (or even $\Theta(n)$) LWE samples, one is able to construct 'fresh' samples *suitable* for BKW.

As for lattice-based methods, BDD decoding is a two-phase algorithm: first, a basis for $\Lambda(\mathbf{A}^{\mathbf{t}})$ is preprocessed via BKZ-reduction to obtain a guarantee on the quality of the output basis (the length of the Gram-Schmidt vectors). With this, we form a search space for the second phase, where we enumerate candidates for the error-vector $\mathbf{e}$ within this search space. Among various ways to enumerate, we start with the greedy approach: Babai's `NearestPlane` algorithm ([11]), then consider its extension, Linder-Peikert `NearestPlanes` ([32]), and finally, pruning strategies ([17], [33]) applied to LWE. The algorithms differ in the shape of the search space and thus, the enumerated candidates are different. Our analysis reveals that for all these techniques, the quantity

$\Upsilon(\textsc{Enum}) = \frac{T(\textsc{Enum})}{P_{\text{succ}}(\textsc{Enum})}$ for the second phase takes the same value in the leading-order term, including the constant in front. Of course, the 'real' trade-off of the whole LWE attack is $\Upsilon(\mathsf{BDD}) = \frac{T(\mathtt{BKZ})+T(\textsc{Enum})}{P_{\text{succ}}(\mathsf{BDD})}$, but as we show below, the second phase – enumeration – dominates over the reduction phase. Now we give a high-level idea of the mentioned enumeration algorithms, where we focus on their geometric meaning. A BDD instance $(\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\}, \mathbf{t} \in \mathscr{L}(\mathbf{B}) + \mathbf{e})$ with a promise on $\|\mathbf{e}\|$ is received as input.

Babai's `NearestPlane`**.** A recursive description of this algorithm is convenient for our later analysis. Given a target $\mathbf{t} \in \mathbb{Z}^m$ and an m-dimensional lattice[3] $\mathscr{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$, we search for $c_m \in \mathbb{Z}$ such that the hyperplane $U_m = c_m\widetilde{\mathbf{b}}_m + \text{Span}(\mathbf{b}_1, \ldots, \mathbf{b}_{m-1})$ is the closest to $\mathbf{t}$. In other words, U_m is the span of the closest translate of the $(m-1)$-dimensional lattice $\mathscr{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{m-1})$. We save this translate (storing c_m), set $\mathbf{t}'$ as the projection of $\mathbf{t}$ on $\text{Span}(\mathbf{b}_1, \ldots, \mathbf{b}_{m-1})$ and call the algorithm recursively with new lower-dimensional target $\mathbf{t}'$ and $\mathscr{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{m-1})$. After m recursive calls, we trace back the translates c_i's and the lattice-vector $\sum_{i=1}^{m} c_i\mathbf{b}_i$ is returned. In Figure 2a, a 2-dimensional example is shown: the hyperplane U spans the closest translate of $\mathscr{L}(\mathbf{b}_1)$ to $\mathbf{t}^{(m)}$. The algorithm is clearly polynomial-time in m, but how can we guarantee that the returned lattice-vector is indeed the closest?

It is easy to verify that the algorithm succeeds if the error vector of the BDD instance lies in $\mathscr{P}_{1/2}(\widetilde{\mathbf{B}})$, the fundamental parallelepiped of the Gram-Schmidt basis $\widetilde{\mathbf{B}}$. Indeed, the distance to the closest translate on each call is bounded by $\frac{1}{2}\|\widetilde{\mathbf{b}}_i\|$. Thus, we stay inside $\mathscr{P}_{1/2}(\widetilde{\mathbf{B}})$ during the execution. How likely is it that the original $\mathbf{e}$ is in $\mathscr{P}_{1/2}(\widetilde{\mathbf{B}})$? This depends on the quality of the input basis $\mathbf{B}$, i.e. the length of the Gram-Schmidt vectors. As we have a guarantee on $\|\mathbf{e}\|$ and the quality of $\mathbf{B}$, in Sect. 4.1 we estimate the success probability of Babai's algorithm, and hence, the ratio $\frac{T(\textsc{Babai})}{P_{\text{succ}}(\textsc{Babai})}$.

Lindner-Peikert `NearestPlanes`**.** The success probability of Babai's algorithm is low, in fact, super-exponentially low for small choices of block size β. Roughly, the length of the error-vector output by Babai's algorithm can be as large as $\sum_{i=1}^{m} \|\widetilde{\mathbf{b}}_i\|^2$, and so might be even larger than $\mathbf{b}_1$, which contradicts the BDD promise. An extension of Babai's algorithm was proposed in [32], where the authors suggest to consider not only the closest translate, but rather several close ones on each recursive call. Thus, we have several candidate solutions in the end. The approach takes into account the skewness of the input basis: taking a further hyperplane in one call might result in much closer hyperplanes on subsequent calls, thereby decreasing the overall error-length. This is illustrated in Figure 2c, where the 3 closest translates of $\mathscr{L}(\mathbf{b}_1)$ are chosen and the solution lies on a further translate. The number of hyperplanes $U_i = c_i\widetilde{\mathbf{b}}_i + \text{Span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ we choose on level i depends on the length of $\widetilde{\mathbf{b}}_i$: the shorter this vector is, the more skewed the lattices in this dimension is, the more hyperplanes we should choose to offset the error. Following [32], by d_i we denote the number of different c_i's on ith level. The resulting search-space is then a stretched fundamental parallelepiped $\mathscr{P}_{1/2}(\widetilde{\mathbf{B}} \cdot \mathbf{D})$, where $\mathbf{D}$ is the diagonal matrix of all d_i's. Babai's algorithm is the special case with $\mathbf{D} = \mathbf{I}$.

[3] The target and the lattice do not have to agree on dimension in general, but for LWE this is the case (we project $\mathbf{t}$ onto $\text{Span}(\mathscr{L}(\mathbf{B}))$ otherwise and work with the projection).

What can we say now about the running-time/success probability? For the latter, we surely can guarantee a constant success probability by making the d_i's sufficiently large. In Sect. 4.2, we estimate the running time for the case of constant success probability.

Length-pruning and variations. The choices of hyperplane(s) in `NearestPlane(s)` *do not* depend on the error-length we have accumulated so far: we hope that our *currently* chosen projection (target) will have relatively close hyperplanes in the subsequent recursive calls contributing to the error-length as little as possible. In other words, expressing the output error-vector via the Gram-Schmidt basis, $\mathbf{e}' = \sum_{i=1}^{m} e'_i \frac{\mathbf{b}_i}{\|\widetilde{\mathbf{b}}_i\|}$, we bound the coordinates $|e'_i|$ individually.

Algorithms that put constraints on the so far accumulated *total error-length* while choosing a hyperplane, so-called length-pruning strategies, were proposed for SVP in [45], extensively studied in [17] and adapted to BDD in [33]. The bound R_i that we impose on the accumulated total length at the ith level is specific to the length-pruning strategy under consideration.

For instance, taking into account the Gaussian nature of LWE-error, one can use tail-bounds for Gaussians and estimate the final error length as $R = \Theta(s\sqrt{m})$. So we could use $R_i = R$ as a trivial bound, having a spherical search space, which guarantees constant success probability. We refer to this as *spherical pruning*. More interesting is setting $R_i = (\frac{m-i+1}{m})^{\frac{1}{2}} \cdot R$ as bounds on the error-length on the ith level (counting i downwards). This case is called *linear pruning* ([45]). Furthermore, instead of having a linear decay, one can think of other bounding functions. [17] considers various choices for R_i and analyzes the running-time/success probability ratio $\Upsilon(\text{ENUM})$ of these algorithms by comparing $\Upsilon(\text{ENUM})$ with the corresponding ratio of spherical pruning.

In Chapter 4.3, we take a more general approach: we consider generalized pruning algorithms that put bounds on the current $|e'_i|$, where the bound arbitrarily depends on the already accumulated e_j's for $j > i$. This covers Babai's, the Lindner-Peikert algorithm, as well as length-pruning strategies. We then give conditions that a reasonable pruning strategy should meet and analyze the trade-off $\Upsilon = \frac{T(\text{ENUM})}{P_{\text{succ}}(\text{ENUM})}$. We show that Υ is asymptotically the same for *any* reasonable generalized pruning strategy.

4 LWE Decoding: General Strategies

In this section, we consider algorithms for solving the LWE problem via bounded distance decoding. In this approach, we first compute a reduced basis for $\Lambda(\mathbf{A}^{\mathrm{t}})$ (the reduction phase) and then find a close lattice point to $\mathbf{t} = \mathbf{A}^{\mathrm{t}}\mathbf{s} + \mathbf{e} \bmod q$ in the decoding phase. The more work we put into basis reduction, the easier the decoding phase will be, so there is a trade-off between the two phases. Since we consider several different algorithms for the decoding phase, we first analyze each decoding algorithm without considering the reduction phase (but on input bases that follow GSA) in Sect. 4.1–4.3 and then discuss the trade-off in Sect. 4.4 for all our algorithms simultaneously. We start by Babai's `NearestPlane` algorithm from [11].

4.1 Babai's `NearestPlane(s)` algorithm

Suppose we are a given a shift[4] $\mathbf{x} \in \mathbb{Q}^m$ and a basis $\mathbf{B} = \mathbf{B}^{(m)} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\} \in \mathbb{Z}^m$ for the shifted lattice $\mathbf{x} + \mathscr{L}(\mathbf{B}^{(m)})$ as well as a target point $\mathbf{t} = \mathbf{x} + \mathbf{v} + \mathbf{e} \in \mathbf{x} + \operatorname{Span}\mathscr{L}(\mathbf{B})$. In the context of LWE decoding, the shift is $\mathbf{x} = 0$ in the inital call and we know that $\mathbf{e}$ is small. Our task is to recover $\mathbf{x} + \mathbf{v}$ or, equivalently, the error vector $\mathbf{e}$. Babai's algorithm for this works as follows: we can write $\mathbf{x} + \mathscr{L}(\mathbf{B}^{(m)})$ as

$$\mathbf{x} + \mathscr{L}(\mathbf{B}^{(m)}) = \bigcup_{i \in \mathbb{Z}} \mathbf{x} + i\mathbf{b}_m + \mathscr{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{m-1}).$$

$\mathbf{x} + i\mathbf{b}_m + \mathscr{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{m-1}) \subset U_i$ is contained in the $m-1$-dimensional hyperplane

$$U_i := \left\{ \mathbf{y} \in \mathbb{R}^m \;\middle|\; \left\langle \mathbf{y}, \frac{\widetilde{\mathbf{b}}_m}{\|\widetilde{\mathbf{b}}_m\|^2} \right\rangle = i + \left\langle \mathbf{x}, \frac{\widetilde{\mathbf{b}}_m}{\|\widetilde{\mathbf{b}}_m\|^2} \right\rangle \right\}. \qquad \text{(cf. Fig. 2a)}$$

Babai's algorithm orthogonally projects $\mathbf{t} = \mathbf{t}^{(m)}$ onto the U_i that is closest to $\mathbf{t}^{(m)}$ to obtain $\mathbf{t}^{(m-1)}$ and then recursively solves the problem for $\mathbf{t}^{(m-1)}$ and the shifted sublattice $(\mathbf{x} + i\mathbf{b}_m) + \mathscr{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{m-1})$. Formally, we obtain the following algorithm:

Algorithm 1 Babai's `NearestPlane` $(\mathbf{B}, \mathbf{x}, \mathbf{t}, \mathbf{e}')$

Input: $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{Z}^{m \times k}, \mathbf{x} \in \mathbb{Q}^m, \mathbf{t} \in \mathbf{x} + \operatorname{Span}\mathbf{B}, \mathbf{e}' \in \mathbb{Q}^m$ ($\mathbf{e}' = \mathbf{x} = 0$ in inital call)
Output: $\mathbf{v} \in \mathbf{x} + \mathscr{L}(\mathbf{B})$ close to $\mathbf{t}$ and $\mathbf{e}' = \mathbf{t} - \mathbf{v}$ corresponding error vector

1: $\mathbf{x}^{(k)} \leftarrow \mathbf{x}, \mathbf{t}^{(k)} \leftarrow \mathbf{t}, \mathbf{e}'^{(k)} \leftarrow \mathbf{e}'$. Let $\widetilde{\mathbf{B}} \leftarrow \mathrm{GSO}(\mathbf{B})$. ▷ For notational purposes
2: **if** $k = 0$ **then return** $(\mathbf{x}, \mathbf{e}')$
3: Compute $u_{\text{old}}^{(k)} \leftarrow \left\langle \mathbf{t}^{(k)}, \frac{\widetilde{\mathbf{b}}_k}{\|\widetilde{\mathbf{b}}_k\|^2} \right\rangle$
4: Choose $u_{\text{new}}^{(k)} = \left\langle \mathbf{x}^{(k)}, \frac{\widetilde{\mathbf{b}}_k}{\|\widetilde{\mathbf{b}}_k\|^2} \right\rangle + i^{(k)}$ closest to $u_{\text{old}}^{(k)}$ with $i^{(k)} \in \mathbb{Z}$.
5: $\mathbf{x}^{(k-1)} \leftarrow \mathbf{x}^{(k)} + i^{(k)}\mathbf{b}_k$ ▷ $\mathbf{x}^{(k-1)} + \mathscr{L}(\mathbf{B}^{(k-1)})$ is nearest plane
6: $\mathbf{e}'^{(k-1)} \leftarrow \mathbf{e}'^{(k)} + (u_{\text{old}}^{(k)} - u_{\text{new}}^{(k)})\widetilde{\mathbf{b}}_k$, $\mathbf{t}^{(k-1)} = \mathbf{t}^{(k)} - (u_{\text{old}}^{(k)} - u_{\text{new}}^{(k)})\widetilde{\mathbf{b}}_k$ ▷ Project onto this plane
7: **return** `NearestPlanes`$((\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}), \mathbf{x}^{(k-1)}, \mathbf{t}^{(k-1)}, \mathbf{e}'^{(k-1)})$

For notational consistency with later algorithms, the argument $\mathbf{e}'$ keeps track of the error vector accumulated so far and is 0 in the initial call. Note that the algorithm constructs the error vector $\mathbf{e}'$ coordinate-wise (wrt. the Gram-Schmidt basis $\widetilde{\mathbf{B}}$), starting from $\widetilde{\mathbf{b}}_m$.

Analysis. Babai's `NearestPlanes` algorithm runs in polynomial time. In the context of LWE decoding, we want that $\mathbf{e}'$ output by the algorithm equals the LWE noise $\mathbf{e}$. Write $\mathbf{e} = \sum_k e_k \frac{\widetilde{\mathbf{b}}_k}{\|\widetilde{\mathbf{b}}_k\|}$ in the Gram-Schmidt basis. We have $\mathbf{e} = \mathbf{e}'$ if all the algorithm's choices of nearest planes are the correct ones, which happens whenever $|e_k| < \frac{1}{2}\|\widetilde{\mathbf{b}}_k\|$ for all k, i.e. if $\mathbf{e}$ is in the interior of $\mathscr{P}_{1/2}(\widetilde{\mathbf{B}})$. The algorithm fails if $|e_k| > \frac{1}{2}\|\widetilde{\mathbf{b}}_k\|$ for any k.[5] For the analysis, we approximate the discrete Gaussian noise $\mathbf{e}$ by a *continuous* one, so the e_k are independent Gaussians with parameter s. For our parameters, the

[4] This is equivalent to the problem for target vector $\mathbf{t} - \mathbf{x}$ and without shift. We use shifts to write the algorithms in a cleaner way via recursion, where shifts will appear in the recursive calls.

[5] On the boundary of $\mathscr{P}_{1/2}(\widetilde{\mathbf{B}})$, it depends on how equally close hyperplanes are handled in line 4 of Alg. 1; this case will not affect our analysis.

case of interest is $\|\widetilde{\mathbf{b}}_m\| \ll s \ll \|\widetilde{\mathbf{b}}_1\|$: in the first steps of the algorithm, we have $s \gg \|\widetilde{\mathbf{b}}_m\|, \|\widetilde{\mathbf{b}}_{m-1}\|, \ldots$, which contributes to a superexponentially small success probability. The $\|\widetilde{\mathbf{b}}_k\|$ increase geometrically (under GSA) from $\|\widetilde{\mathbf{b}}_m\|$ to $\|\widetilde{\mathbf{b}}_1\|$. At some intermediate critical k^*, we have $s \approx \|\widetilde{\mathbf{b}}_{k^*}\|$ and the subsequent steps do not contribute much to the failure probability. More precisely:

Lemma 1. *Let the sequence $\|\widetilde{\boldsymbol{b}}_1\| > \ldots > \|\widetilde{\boldsymbol{b}}_n\|$ be geometrically decreasing with decay rate $\|\widetilde{\boldsymbol{b}}_i\|/\|\widetilde{\boldsymbol{b}}_{i+1}\| = \delta^2 > 1$. Let $e_1, \ldots, e_n$ be independent continuous Gaussians with density $\frac{1}{s}\exp(-\frac{\pi x^2}{s^2})$. Let $p_i := \Pr\big[|e_i| < \|\widetilde{\boldsymbol{b}}_i\|\big]$. Then*

- *If $\|\widetilde{\boldsymbol{b}}_n\| > s(\log n)^{\frac{1}{2}+\varepsilon}$ for fixed $\varepsilon = \Theta(1), \varepsilon > 0$, then $\prod_i p_i = 1 - o(1)$.*
- *If $\|\widetilde{\boldsymbol{b}}_n\| = s$, then $\prod_i p_i = 2^{-\mathcal{O}(n)}$.*
- *If $\|\widetilde{\boldsymbol{b}}_1\| = s$, then $\prod_i p_i = 2^{-\mathcal{O}(n)} \cdot 2^n \delta^{-n(n-1)}$.*

Proof. By Eq. (1), $1 - p_i$ is superpolynomially small if $\|\widetilde{\mathbf{b}}_i\| > s(\log n)^{\frac{1}{2}+\varepsilon}$. The first statement then follows by a union bound. The second statement is trivial, as we have $\min_i p_i = \Omega(1)$. For the third statement, we estimate for $\|\widetilde{\mathbf{b}}_i\| < s$

$$p_i = \int_{-\|\widetilde{\mathbf{b}}_i\|}^{+\|\widetilde{\mathbf{b}}_i\|} \frac{1}{s} e^{-\pi\frac{x^2}{s^2}} \mathrm{d}x = \Theta(1) \int_{-\|\widetilde{\mathbf{b}}_i\|}^{+\|\widetilde{\mathbf{b}}_i\|} \frac{1}{s} \mathrm{d}x = \Theta(1) \frac{\|\widetilde{\mathbf{b}}_i\|}{s/2}.$$

So $\prod_i p_i = 2^{-\mathcal{O}(n)} \frac{\prod \|\widetilde{\mathbf{b}}_i\|}{(s/2)^n} = 2^{-\mathcal{O}(n)} \frac{2^n \left(\|\widetilde{\mathbf{b}}_1\| \|\widetilde{\mathbf{b}}_n\|\right)^{n/2}}{s^n} = 2^{-\mathcal{O}(n)} \frac{2^n \|\widetilde{\mathbf{b}}_n\|^{n/2}}{s^{n/2}} = 2^{-\mathcal{O}(n)} 2^n \delta^{-n(n-1)}$.

This implies the following theorem for Babai's `NearestPlanes` algorithm:

Theorem 2. *Under the Geometric Series Assumption (GSA) on a $\beta = \Theta(n)$ reduced-basis that arises from $m = (c_m + o(1))n$ LWE samples with parameters $(n, q = \mathcal{O}(n^{c_q}), s = \mathcal{O}(n^{c_s}))$ for c_m and $c_s < c_q$ constants, Babai's `NearestPlanes` algorithm solves the search-LWE problem in polynomial time with success probability*

$$P_{succ}(\textsc{Babai}) = \begin{cases} 2^{-\frac{1}{2}\left(\frac{m}{2\beta} - c_q + c_m^{-1} c_q + c_s\right)^2 (1+o(1))\beta \log \beta}, & \text{if } \frac{m}{2\beta} - c_q + c_m^{-1} c_q + c_s > 0 \\ 1 - o(1), & \text{if } \frac{m}{2\beta} - c_q + c_m^{-1} c_q + c_s < 0, \end{cases}$$

if we assume that the LWE error follows a continuous Gaussian distribution.

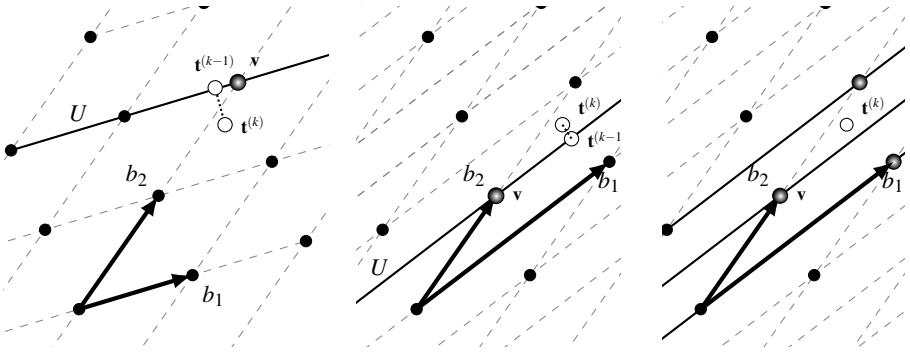
Note that the two cases in the theorem relate to whether $\|\mathbf{b}_1\|$ is larger or smaller than s.

Proof. Under GSA, we have $\|\widetilde{\mathbf{b}}_i\| = \beta^{\frac{m}{2\beta}} q^{1 - c_m^{-1}} \delta^{-2i}$ with $\delta = \beta^{\frac{1}{2\beta}}$. A simple computation shows that $\|\widetilde{\mathbf{b}}_{k^*}\| = n^{c_s}$ for the critical $k^* = \beta(\frac{m}{2\beta} + c_q - c_q c_m^{-1} - c_s)$. Consequently,

$$m - k^* = \beta\left(\tfrac{m}{2\beta} - c_q + \tfrac{n}{m} c_q + c_s\right). \tag{4}$$

If $\frac{m}{2\beta} - c_q + c_m^{-1} c_q + c_s > 0$, we actually have $k^* > m$ and $\|\widetilde{\mathbf{b}}_m\| > s \cdot \mathrm{poly}(n)$. The success probability is $1 - o(1)$ by the first part of Lemma 1. If $\frac{m}{2\beta} - c_q + c_m^{-1} c_q + c_s < 0$, by the third part of that lemma, we have

$$P_{\mathrm{succ}}(\textsc{Babai}) = 2^{-\mathcal{O}(m)} 2^{m-k^*} \delta^{(m-k^*)^2} = 2^{-\frac{1}{2}\left(\frac{m}{2\beta} - c_q + \frac{n}{m} c_q + c_s + o(1)\right)^2 \beta \log \beta}.$$



(a) Babai's `NearestPlane` Algorithm for a good basis. A target point $t^{(k)}$ is projected on a closest hyperplane $2\mathbf{b}_2 + \mathrm{Span}(\mathbf{b}_1)$. The recursive call for the chosen one-dimensional subspace U (thicker) projects on the closest zero-dimensional hyperplane - the shaded lattice point $\mathbf{v}$.

(b) Babai's `NearestPlane` Algorithm for a bad basis. Now the closest hyperplane is $\mathbf{b}_2 + \mathrm{Span}(\mathbf{b}_1)$. The chosen one-dimensional subspace U has changed, so has the vector $\mathbf{v}$. Obviously, this lattice vector is not the solution.

(c) In Lindner-Peikert's generalization, we stretch the bad basis case by setting $d_2 = 3$ in directions $2\mathbf{b}_2, \mathbf{b}_2, 0\mathbf{b}_2$, and recurse on each. The process collects all shaded points and therefore also the closest one.

Fig. 2: `NearestPlane(s)` Algorithms

4.2 Lindner-Peikert `NearestPlanes` Algorithm

Babai's algorithm is characterized by its *search region* $V_{\text{Babai}} = \mathscr{P}_{1/2}(\widetilde{\mathbf{B}})$. Indeed, it returns the unique[6] $\mathbf{v} \in \mathscr{L}(\mathbf{B})$ with $\mathbf{t} \in \mathbf{v} + \mathscr{P}_{1/2}(\widetilde{\mathbf{B}})$. Therefore, the more orthogonal the input basis $\mathbf{B}$ is, the better $\mathbf{v}$ approximates the lattice-vector closest to $\mathbf{t}$ (in Figure 2a the basis vectors are fairly orthogonal). However, the procedure performs far worse if a given basis is 'long and skinny' (Figure 2b) and the error increases as the dimension grows. In terms of the LWE decoding problem this means that Babai's `NearestPlane` will solve the search LWE-problem iff the error vector $\mathbf{e}$ lies in $\mathscr{P}_{1/2}(\widetilde{\mathbf{B}})$. For typical parameters, this is rather unlikely since the last Gram-Schmidt vectors in a `BKZ` reduced basis are fairly short.

To address this problem, Lindner and Peikert suggested to choose several ($d_i \geq 1$) close hyperplanes in the ith level of the recursion (Figure 2c). Geometrically, this means that we stretch the elongated parallelepiped $\mathscr{P}_{1/2}(\widetilde{\mathbf{B}})$ to a cube-like shape by increasing the last and therefore short Gram-Schmidt vectors. In the end, we have $d_m \cdots d_1$ candidate solutions to check. Formally, the algorithm works as follows:

[6] This can be made to hold true even if there are two equally close hyperplanes in the algorithm; these cases do not affect our analysis.

Algorithm 2 Lindner-Peikert's `NearestPlanes` $(\mathbf{B}, \mathbf{x}, \mathbf{t}, \mathbf{e}')$

Input: $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{Z}^{m \times k}, \mathbf{x} \in \mathbb{Q}^m, \mathbf{t} \in \mathbf{x} + \operatorname{Span}\mathbf{B}, \mathbf{e}' \in \mathbb{Q}^m$ ($\mathbf{e}' = \mathbf{x} = 0$ in inital call)
Output: A set of pairs $(\mathbf{v}, \mathbf{e}')$ with $\mathbf{v} \in \mathbf{x} + \mathscr{L}(\mathbf{B})$ and $\mathbf{e}' = \mathbf{t} - \mathbf{v}$ corresponding error vector

1: $\mathbf{x}^{(k)} \leftarrow \mathbf{x}, \mathbf{t}^{(k)} \leftarrow \mathbf{t}, \mathbf{e}'^{(k)} \leftarrow \mathbf{e}'$. Let $\widetilde{\mathbf{B}} \leftarrow \mathrm{GSO}(\mathbf{B})$. ▷ For notational purposes
2: **if** $k = 0$ **then return** $\{(\mathbf{x}, \mathbf{e}')\}$
3: Compute $u_{\mathrm{old}}^{(k)} \leftarrow \left\langle \mathbf{t}^{(k)}, \frac{\widetilde{\mathbf{b}}_k}{\|\widetilde{\mathbf{b}}_k\|^2} \right\rangle$
4: Let $u_j^{(k)} = \left\langle \mathbf{x}^{(k)}, \frac{\widetilde{\mathbf{b}}_k}{\|\widetilde{\mathbf{b}}_k\|^2} \right\rangle + i_j^{(k)}$ for $i_j^{(k)} \in \mathbb{Z}, j = 1, \ldots, d_k$ be the d_k closest numbers to $u_{\mathrm{old}}^{(k)}$.
5: Let $\mathbf{x}_j^{(k-1)} \leftarrow \mathbf{x}^{(k)} + i_j^{(k)} \mathbf{b}_k$ for $1 \le j \le d_k$ ▷ $\mathbf{x}_j^{(k-1)} + \mathscr{L}(\mathbf{B}^{(k-1)})$ are the d_k nearest planes
6: $\mathbf{e}_j'^{(k-1)} \leftarrow \mathbf{e}'^{(k)} + (u_{\mathrm{old}}^{(k)} - u_j^{(k)})\widetilde{\mathbf{b}}_k, \mathbf{t}_j^{(k-1)} = \mathbf{t}^{(k)} - (u_{\mathrm{old}}^{(k)} - u_j^{(k)})\widetilde{\mathbf{b}}_k,$ ▷ Project onto them
7: **return** $\bigcup_j$ `NearestPlanes`$((\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}), \mathbf{x}_j^{(k-1)}, \mathbf{t}_j^{(k-1)}, \mathbf{e}_j'^{(k-1)})$

Analysis. Our search region is now extended to $V_{\mathrm{LP}} = \mathscr{P}_{1/2}(\widetilde{\mathbf{B}} \cdot \mathbf{D})$, which, as estimated in [32], amplifies the success probability for LWE decoding to

$$P_{\mathrm{LP}} := \Pr[\mathbf{e} \in \mathscr{P}_{1/2}(\widetilde{\mathbf{B}} \cdot \mathbf{D})] = \prod_{i=1}^{m} \Pr\left[\left|\left\langle \mathbf{e}, \widetilde{\mathbf{b}}_i \right\rangle\right| < \tfrac{d_i \cdot \|\widetilde{\mathbf{b}}_i\|^2}{2}\right] = \prod_{i=1}^{m} \operatorname{erf}\left(\tfrac{d_i \|\widetilde{\mathbf{b}}_i\| \sqrt{\pi}}{2s}\right),$$

where $\mathbf{D}$ is the diagonal matrix composed of the d_i's, and $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$. Here, like in Sect. 4.2, we estimate the discrete Gaussian error $\mathbf{e}$ by a continuous one. We wish to set the parameters d_i such that the success probability is at least constant. It follows from Eq. (1) that if $\min_{1 \le i \le m} \frac{d_i \|\mathbf{b}_i\|}{s} = \omega(\sqrt{\log m})$, then $P_{\mathrm{LP}} = 1 - o(1)$. Conversely, if we have $\min_{1 \le i \le m} \frac{d_i \|\mathbf{b}_i\|}{s} = o(1)$, then $P_{\mathrm{LP}} = o(1)$. So we set $d_i = \left\lceil \frac{s \cdot (\log m)^\alpha}{\|\widetilde{\mathbf{b}}_i\|} \right\rceil$ for sufficiently large $\alpha > 1/2$ (which will not affect the asymptotic running time anyway). Note that for our parameter choices, $\|\widetilde{\mathbf{b}}_1\|$ will be much larger than s, so the first d_i's in the algorithm are all equal to 1 and the sequence of the d_i's has the form $(1, 1, \ldots, 1, 2, \ldots)$.

Let us turn our attention to the running time. The recursive calls to the Linder-Peikert algorithm have the structure of a rooted tree, where the root corresponds to the initial call and every node calls its d_k children. The leaves correspond to the candidate solutions we need to check in the end. Note that every node at level k (the root has level m and the leaves have level 0) corresponds to a partial solution, where we already fixed the last $m - k$ coefficients of $\mathbf{e}'$ (wrt. the $\widetilde{\mathbf{B}}$-basis) and of $\mathbf{x}$ (wrt. the $\mathbf{B}$-basis). Let N_k be the number of nodes at level k and N be the total number of nodes. Clearly, $N_k = \prod_{i=k+1}^{m} d_i$ and $N = \sum_{k=0}^{m} N_k$. As the running time per node is polynomial, estimating the running time amounts to estimating N. For this, we have the following result.

Theorem 3. *Under the Geometric Series Assumption (GSA) on a $\beta = \Theta(n)$-reduced basis that arises from $m = (c_m + o(1))n$ LWE samples with parameters $(n, q = \mathscr{O}(n^{c_q}), s = \mathscr{O}(n^{c_s}))$ for c_m and $c_s < c_q$ constants, `NearestPlanes` with our choice of d_i's solves search-LWE problem with success probability $1 - o(1)$ in time*

$$T_{LP} = \mathrm{poly}(n) N = \begin{cases} 2^{\frac{1}{2}\left(\frac{m}{2\beta} - c_q + c_m^{-1} c_q + c_s\right)^2 (1 + o(1))\beta \log \beta}, & \text{if } \frac{m}{2\beta} - c_q + c_m^{-1} c_q + c_s > 0 \\ \mathrm{poly}(n), & \text{if } \frac{m}{2\beta} - c_q + c_m^{-1} c_q + c_s < 0 \end{cases}$$

and polynomial memory (using depth-first search), if we assume that the LWE error follows a continuous Gaussian distribution.

Proof. If $\frac{m}{2\beta} - c_q + c_m^{-1} c_q + c_s < 0$, we have $\|\widetilde{\mathbf{b}}_m\| > s \cdot \mathrm{poly}(m)$, so for sufficiently large m, all $d_i = 1$ and the result follows from Thm. 2. So consider the case $\frac{m}{2\beta} - c_q + c_m^{-1} c_q + c_s > 0$. Since $N_k \leq N_0$ for all k, we have $N_0 \leq N \leq (m+1) \cdot N_0$. So up to polynomial factors, the running time is given by $N_0 = \prod d_i$. Let the critical k^* be maximal, s.t. $\|\widetilde{\mathbf{b}}_{k^*}\| > s$. By Eq. (4), $\frac{m-k^*}{\beta} = \frac{m}{2\beta} - c_q + \frac{n}{m} c_q + c_s$. We compute for $N_0 = \prod_i d_i$:

$$\prod_{i=k^*+1}^{m} \frac{s}{\|\widetilde{\mathbf{b}}_i\|} \leq \prod_{i=1}^{m} \left\lceil \frac{s \cdot (\log m)^\alpha}{\|\widetilde{\mathbf{b}}_i\|} \right\rceil = \prod_i d_i = N_0 \qquad \text{and}$$

$$N_0 \leq (1 + (\log m)^\alpha)^m \prod_{i=1}^{m} \left\lceil \frac{s}{\|\widetilde{\mathbf{b}}_i\|} \right\rceil \leq (1 + (\log m)^\alpha)^m 2^{m-k^*} \prod_{i=k^*+1}^{m} \frac{s}{\|\widetilde{\mathbf{b}}_i\|}.$$

We already computed (the inverse of) $\prod_{i=k^*+1}^{m} \frac{s}{\|\widetilde{\mathbf{b}}_i\|}$ in the analysis of Babai's algorithm (cf. Lemma 1 and Thm. 2), so

$$\prod_{i=k^*+1}^{m} \frac{s}{\|\widetilde{\mathbf{b}}_i\|} = 2^{\left(\frac{(m-k^*)^2}{2\beta^2} + o(1)\right)\beta \log \beta},$$

which is exactly what we want. The error term $(1 + (\log m)^\alpha)^m 2^{m-k^*} = 2^{\mathcal{O}(m \log\log m)}$ only contributes to the $o(1)$-term.

4.3 Generalized Pruning Strategies

In Babai's or Lindner and Peikert's algorithm, at every node at level k in the search tree, we have already fixed the coordinates $e'_m, \ldots, e'_{k+1}$ of the output error vector $\mathbf{e}' = \sum_i e'_i \frac{\widetilde{\mathbf{b}}_i}{\|\widetilde{\mathbf{b}}_i\|}$ (in the Gram-Schmidt basis). These coordinates are contained in the argument $\mathbf{e}'^{(k)} = \sum_{i=k+1}^{m} e'_i \frac{\widetilde{\mathbf{b}}_i}{\|\widetilde{\mathbf{b}}_i\|}$ that we pass on during recursion. In particular, we already know that the final error vector will have length at least $\|\mathbf{e}'^{(k)}\|^2$. We then recurse on exactly d_k children (i.e. the d_k closest hyperplanes), where $d_k = 1$ for Babai and $d_k \geq 1$ for Lindner-Peikert. But actually, it makes sense to make the number of children variable, depending on the error vector accumulated so far. If $\mathbf{e}'^{(k)}$ is very small, the node is more likely to lead to the correct solution, hence we should choose more children, whereas if $\|\mathbf{e}'^{(k)}\| \gg \sqrt{m}s$, this node will probably not lead to the correct solution, so we might choose no children at all. We now generalize Babai's / Lindner and Peikert's algorithm to allow for arbitrary dependency of the number of children on the error accumulated so far. For any (efficiently computable) family of bounding functions $B^{(k)} \colon \mathbb{Q}_{\geq 0}^{m-k} \to \mathbb{Q}_{\geq 0}$, $1 \leq k \leq m$, we consider the following generalized pruning algorithm `GenPruning`:

Algorithm 3 Generalized Pruning Algorithm $\texttt{GenPruning}(\mathbf{B},\mathbf{x},\mathbf{t},\mathbf{e}')$ for a family of bounding functions $B^{(k)}$

Input: $\mathbf{B}=(\mathbf{b}_1,\dots,\mathbf{b}_k)\in\mathbb{Z}^{m\times k},\mathbf{x}\in\mathbb{Q}^m,\mathbf{t}\in\mathbf{x}+\operatorname{Span}\mathbf{B},\mathbf{e}'\in\mathbb{Q}^m$ ($\mathbf{e}'=\mathbf{x}=0$ in inital call)
Output: A set of pairs $(\mathbf{v},\mathbf{e}')$ with $\mathbf{v}\in\mathbf{x}+\mathscr{L}(\mathbf{B})$ and $\mathbf{e}'=\mathbf{t}-\mathbf{v}$ corresponding error vector

1: $\mathbf{x}^{(k)}\leftarrow\mathbf{x},\mathbf{t}^{(k)}\leftarrow\mathbf{t},\mathbf{e}'^{(k)}\leftarrow\mathbf{e}'$. Let $\widetilde{\mathbf{B}}\leftarrow\mathrm{GSO}(\mathbf{B})$. ▷ For notational purposes
2: **if** $k=0$ **then return** $\{(\mathbf{x},\mathbf{e}')\}$
3: Compute $u_{\text{old}}^{(k)}\leftarrow\langle\mathbf{t}^{(k)},\frac{\widetilde{\mathbf{b}}_k}{\|\widetilde{\mathbf{b}}_k\|^2}\rangle$.
4: Let $e'_i=\langle\mathbf{e}'^{(k)},\frac{\mathbf{b}_i}{\|\mathbf{b}_i\|}\rangle$ for $k<i\le m$. ▷ Coefficients of $\mathbf{e}'$
5: Let $D^2_{\max}=B^{(k)}(e'^2_m,\dots,e'^2_{k+1})$ ▷ bound on distance of next hyperplanes
6: Let $u_j^{(k)}=\langle\mathbf{x}^{(k)},\frac{\widetilde{\mathbf{b}}_k}{\|\widetilde{\mathbf{b}}_k\|^2}\rangle+i_j^{(k)}$, $j=1,\dots$ be all possible numbers s.t.
$|u_{\text{old}}^{(k)}-u_j^{(k)}|^2\cdot\|\widetilde{\mathbf{b}}_k\|^2\le D^2_{\max}$ and $i_j^{(k)}\in\mathbb{Z}$.
7: Let $\mathbf{x}_j^{(k-1)}\leftarrow\mathbf{x}^{(k)}+i_j^{(k)}\mathbf{b}_k$ for all j ▷ Consider the nearby planes $\mathbf{x}_j^{(k-1)}+\mathscr{L}(\mathbf{B}^{(k-1)})$
8: $\mathbf{e}_j'^{(k-1)}\leftarrow\mathbf{e}'^{(k)}+(u_{\text{old}}^{(k)}-u_j^{(k)})\widetilde{\mathbf{b}}_k$, $\mathbf{t}_j^{(k-1)}=\mathbf{t}^{(k)}-(u_{\text{old}}^{(k)}-u_j^{(k)})\widetilde{\mathbf{b}}_k$, ▷ Project onto them
9: **return** $\bigcup_j\texttt{GenPruning}((\mathbf{b}_1,\dots,\mathbf{b}_{k-1}),\mathbf{x}_j^{(k-1)},\mathbf{t}_j^{(k-1)},\mathbf{e}_j'^{(k-1)})$

The algorithm recurses on all hyperplanes, s.t. $(e'_k)^2\le B^{(k)}(e'^2_m,\dots,e'^2_{k+1})$. So the search region of `GenPruning` is given by

$$V_{\text{GP}}=\left\{\mathbf{e}'=\sum_i e'_i\frac{\widetilde{\mathbf{b}}_i}{\|\widetilde{\mathbf{b}}_i\|}\;\middle|\;e'^2_k\le B^{(k)}(e'^2_m,\dots,e'^2_{k+1})\text{ for all }k\right\}$$

and the algorithm is successful if the LWE error vector is contained in V_{GP}. `GenPruning` captures what is known as pruned enumeration [45]: in pruned enumeration, we keep only partial candidate solutions where the partial error vectors $\mathbf{e}'^{(k)}$ satisfy $\|\mathbf{e}'^{(k)}\|\le R_k$ for some level-dependent bounds R_k that are defined by the particular pruning strategy. This is achieved by setting

$$B^{(k)}(e'^2_m,\dots,e'^2_{k+1})=R_k^2-\sum_i e'^2_i. \tag{5}$$

For instance, we get the following algorithms for those specific choices of $B^{(k)}$:

- $B^{(k)}(e'^2_m,\dots,e'^2_{k+1})=\left(\frac{\|\widetilde{\mathbf{b}}_k\|}{2}\right)^2$: Babai's algorithm[7].
- $B^{(k)}(e'^2_m,\dots,e'^2_{k+1})=\left(\frac{d_k\|\widetilde{\mathbf{b}}_k\|}{2}\right)^2$: Linder-Peikert algorithm.
- $B^{(k)}(e'^2_m,\dots,e'^2_{k+1})=\Theta(ms^2)-\sum_{i=k+1}^m e'^2_i$: Spherical Pruning.
- $B^{(k)}(e'^2_m,\dots,e'^2_{k+1})=\Theta((m-k-1)s^2)-\sum_{i=k+1}^m e'^2_i$: Linear Pruning.

We also cover the extreme pruning approach of [17] and potential algorithms where $B^{(k)}$ has a more complicated dependency on the individual e'_i's. For technical reasons, we require that $B^{(k)}$ is extended to real-valued arguments in a continuous way, so the algorithm becomes meaningful for error vectors following a continuous Gaussian and we analyze it for such continuous errors. Furthermore, the bounding functions do not get the Gram-Schmidt vectors or their lengths $\|\widetilde{\mathbf{b}}_i\|$ as explicit inputs, but we rather

[7] Again, we ignore the case of equally close hyperplanes for `NearestPlane(s)`.

treat these lengths as a promise. Essentially, this mean that we restrict to enumeration algorithms that are oblivious to the actual geometry of the lattice.

Analysis. We are interested in the ratio of expected running time to success probability $\Upsilon(\texttt{GP}) = \frac{T(\texttt{GP})}{P_{\text{succ}}(\texttt{GP})}$ if the LWE error vector $\mathbf{e}$ follows a continuous Gaussian with parameter s. Let V_k be the search region at level k, i.e.

$$V_k = \left\{ \mathbf{e}' = \sum_{i=k+1}^{m} e'_i \frac{\widetilde{\mathbf{b}}_i}{\|\widetilde{\mathbf{b}}_i\|} \;\middle|\; e'^2_j \leq B^{(j)}(e'^2_m, \ldots, e'^2_{j+1}), k < j \leq m \right\}$$

and N_k be the number of nodes at level k during a run of `GenPruning`. Let P_k be the probability that we retain the correct solution at level k (i.e. we have a partial solution at level k that can be extended to the solution of the search-LWE problem). Note that N_k is the number of points in $\mathbf{e} - V_k$ that belong to the lattice $\pi_k(\mathscr{L}(\mathbf{B}))$, where π_k is the projection onto the orthogonal complement of $\text{Span}(\mathbf{b}_1, \ldots, \mathbf{b}_k)$. So, we expect that

$$N_k \approx \frac{\text{vol}\, V_k}{\|\widetilde{\mathbf{b}}_m\| \cdots \|\widetilde{\mathbf{b}}_{k+1}\|} \tag{6}$$

by what is known as the Gaussian Heuristic. Interestingly enough, we only require Eq. (6) to hold up to a factor $2^{\mathscr{O}(m)}$, which we can prove rigorously for *convex* V_k and taking the expected value (over $\mathbf{e}$) of N_k, using a variant [42] of Minkowski's Convex Body Theorem. We will prove matching lower and upper bounds for $\Upsilon(\texttt{GP})$, where for the upper bound we need to impose some (weak) restrictions on the bounding functions.

Definition 4. *Assume that the $\|\widetilde{\boldsymbol{b}}_i\|$ follow the GSA and that our correct error vector follows a continuous Gaussian with parameter s. Assume that $\|\widetilde{\boldsymbol{b}}_1\| > s > \|\widetilde{\boldsymbol{b}}_m\|$ and let k^* be maximal s.t. $\|\widetilde{\boldsymbol{b}}_{k^*}\| > s$. We call $B^{(k)}$ resp. the associated generalized pruning algorithm* reasonable *for the given parameters if the following conditions are satisfied:*

1. $B^{(k)}(e'^2_m, \ldots, e'^2_{k+1}) + \sum_{i=k+1}^{m} e'^2_i = \mathscr{O}(ms^2)$.
2. $P_{succ}(GP) \geq 2^{-O(m)} P_{k^*}$.
3. $N_k \leq 2^{\mathscr{O}(m)} N_{k^*}$ *for* $k \leq k^*$.
4. $\frac{N_{k-1}}{N_k} = \Omega(1)$ *for* $k \geq k^*$.
5. V_{k^*} *is convex.*

Condition 1 means that we do not consider partial solution where the accumulated error vector up to that level point is already larger than the expected the final error vector. Conditions 2 and 3 tell us that, up to exponential factors, we can find the correct solution at little cost, provided it survived until level k^*. Note that switching to Babai's algorithm from level k^* on will ensure that conditions 2 and 3 hold. Condition 4 tells us that the average number of children nodes at level i is at least constant as long as $\|\widetilde{\mathbf{b}}_i\| < s$. Note that $\|\widetilde{\mathbf{b}}_i\| < s$ implies that we always have $\Omega(1)$ plausible candidate hyperplanes at distance at most s. The convexity property holds for all pruned enumeration strategies that use Eq. (5) and may be replaced by asking that heuristic Eq. (6) holds up to exponential error.

It is not hard to see that Babai's algorithm, Linear Pruning, Spherical Pruning and all pruning algorithms considered in [17] are reasonable. The Lindner-Peikert algorithm is literally a corner case: in the corners of the parallelepiped-shaped search region, condition 1 is (barely) violated, which is why we analyzed it separately.

Theorem 5. *Under the Geometric Series Assumption (GSA) on a $\beta = \Theta(n)$-reduced basis that arises from $m = (c_m + o(1))n$ LWE samples with parameters $(n, q = \mathcal{O}(n^{c_q}), s = \mathcal{O}(n^{c_s}))$ for c_m and $c_s < c_q$ constants, such that $\frac{m}{2\beta} - c_q + c_m^{-1}c_q + c_s > 0$, any reasonable generalized pruning algorithm GP has an expected running time to success probability ratio Υ of*

$$\Upsilon(\mathit{GP}) = \frac{\mathbb{E}[T(\mathit{GP})]}{P_{succ}(\mathit{GP})} = 2^{\frac{1}{2}\left(\frac{m}{2\beta} - c_q + c_m^{-1}c_q + c_s\right)^2 (1+o(1))\beta \log \beta}.$$

Furthermore, if Eq. (6) *holds up to at most an exponential factor then, even if GP is not reasonable, the above is also a* lower bound *for $\Upsilon(\mathit{GP})$.*

Proof. The running time is clearly $\mathrm{poly}(m) \cdot \sum_i N_i$. If the pruning strategy is reasonable, we have

$$P_{\mathrm{succ}}(\mathtt{GP}) = 2^{\mathcal{O}(-m)} P_{k^*} \qquad \text{and} \qquad T(\mathtt{GP}) = \mathrm{poly}(m) \sum_i N_i = 2^{\mathcal{O}(m)} N_{k^*},$$

because $N_i \leq 2^{\mathcal{O}(m)} N_{k^*}$ for all i, no matter whether $i > k^*$ or $i < k^*$. Now, N_{k^*} is the number of points from $\mathbf{e} - V_{k^*}$ in the lattice $\pi_{k^*}(\mathscr{L}(\mathbf{B}))$. Hence, the expected value (over the choice of $\mathbf{e}$) of N_{k^*} is given by

$$\mathop{\mathbb{E}}_{\mathbf{e}}[N_{k^*}] = \sum_{\mathbf{x} \in \pi_{k^*}(\mathscr{L}(\mathbf{B}))} f(\mathbf{x}) \quad \text{where} \quad f(\mathbf{x}) = \int_{V_{k^*}} \frac{1}{s^{k^*}} \exp(-\tfrac{\pi \|\mathbf{y}\|^2}{s^2}) \mathrm{d}\mathbf{y}.$$

Since V_{k^*} is convex, its characteristic function is log-concave. The Gaussian density is log-concave. So f is the convolution of two log-concave functions, hence log-concave itself. In particular, f is centrally symmetric and quasiconcave. It follows by a variant of Minkowsi's Convex Body Theorem due to R. Rado [42] that

$$\mathop{\mathbb{E}}_{\mathbf{e}}[N_{k^*}] = \frac{2^{\pm \mathcal{O}(m)}}{\det(\pi_{k^*}(\mathscr{L}(\mathbf{B})))} \int f(\mathbf{x}) \mathrm{d}\mathbf{x} = \frac{2^{\pm \mathcal{O}(m)} \operatorname{vol} V_{k^*}}{\det(\pi_{k^*}(\mathscr{L}(\mathbf{B})))}. \tag{7}$$

By Condition 1 in Def. 4, we have $1 \geq \exp(-\frac{\pi \mathbf{x}^2}{s^2}) \geq e^{-\mathcal{O}(m)}$ for every $\mathbf{x} \in V_{k^*}$. So

$$\begin{aligned} \frac{\mathbb{E}[T]}{P_{\mathrm{succ}}(\mathtt{GP})} &= \frac{2^{\pm \mathcal{O}(m)} \frac{\operatorname{vol} V_{k^*}}{\|\widetilde{\mathbf{b}}_m\| \cdots \|\widetilde{\mathbf{b}}_{k+1}\|}}{\int_{\mathbf{x} \in V_{k^*}} \frac{1}{s^{m-k^*}} \exp(-\frac{\pi \|\mathbf{x}\|^2}{s^2}) \mathrm{d}\mathbf{x}} = \frac{2^{\pm \mathcal{O}(m)} s^{m-k^*}}{\|\widetilde{\mathbf{b}}_m\| \cdots \|\widetilde{\mathbf{b}}_{k+1}\|} \frac{\int_{\mathbf{x} \in V_{k^*}} 1 \mathrm{d}\mathbf{x}}{\int_{\mathbf{x} \in V_{k^*}} \exp(-\frac{\pi \|\mathbf{x}\|^2}{s^2}) \mathrm{d}\mathbf{x}} \\ &= \frac{2^{\pm \mathcal{O}(m)} s^{m-k^*}}{\|\widetilde{\mathbf{b}}_m\| \cdots \|\widetilde{\mathbf{b}}_{k+1}\|} \frac{\int_{\mathbf{x} \in V_{k^*}} 1 \mathrm{d}\mathbf{x}}{\int_{\mathbf{x} \in V_{k^*}} 1 \mathrm{d}\mathbf{x}} = 2^{\frac{1}{2}\left(\frac{m}{2\beta} - c_q + c_m^{-1}c_q + c_s\right)^2 (1+o(1))\beta \log \beta}. \end{aligned} \tag{8}$$

If the pruning strategy is not reasonable, we still have $P_{\mathrm{succ}}(\mathtt{GP}) \leq P_{k^*}$, $T \geq \mathrm{poly}(m) N_{k^*}$ and $1 \geq \exp(-\frac{\pi \mathbf{x}^2}{s^2})$ as trivial bounds. Eq (7) holds by assumption. This is sufficient to prove Eq. (8) with $\geq$ instead of an equality.

4.4 Balancing the reduction and enumeration phases

So far we have been concerned with the quantity $\Upsilon(\textsc{Enum}) = \frac{T(\textsc{Enum})}{P_{\mathrm{succ}}(\textsc{Enum})}$, but, as pointed out in Sect. 3, the BDD attack is actually a two-phase attack, where the enumeration

(phase 2) is performed on a β-reduced basis. Thus, the ratio one should look at is $\Upsilon(\mathsf{BDD}) = \frac{T(\mathtt{BKZ})+T(\mathrm{ENUM})}{P_{\mathrm{succ}}(\mathrm{ENUM})}$. We want to minimize $\Upsilon(\mathsf{BDD})$.

On input β, a lattice reduction algorithm calls as a subroutine an SVP-solver in a sublattice of dimension β. The running time of this solver essentially determines the complexity of the whole reduction. There are two ways to instantiate this SVP solver: the first is due to [25] with super-exponential running time of $2^{\mathcal{O}(\beta \log \beta)}$ (and polynomial space) and algorithms of [39] and [2] that achieve single-exponential complexity $2^{\mathcal{O}(\beta)}$, but requiring exponential storage. These two cases give rise to the following two theorems that state the complexity of the whole BDD attack. For the enumeration phase we consider any reasonable generalized pruning strategy (Def. 4) or the Lindner-Peikert algorithm. We make a distinction between pruning strategies that achieve constant (e.g. Lindner-Peikert, spherical pruning) and arbitrarily small (e.g. Babai, extreme pruning) success probability.

Theorem 6. *With a $\beta = \Theta(n)$-reduced basis reduction running in time $2^{c_{BKZ} \cdot \beta \log \beta}$ and any reasonable generalized pruning algorithm GP (or Lindner-Peikert), the complexity of solving the LWE problem with parameters $(n, q = \mathcal{O}(n^{c_q}), s = \mathcal{O}(n^{c_s}))$ via BDD using the optimal choice of $m = n \cdot (\frac{2c_q}{\sqrt{2c_{BKZ}}+c_q-c_s} + o(1))$ samples is*

$$T(\mathit{BDD}) = \Theta(\Upsilon(\mathit{BDD})) = 2^{\left(c_{BKZ} \cdot \frac{2c_q}{(\sqrt{2c_{BKZ}}+c_q-c_s)^2} + o(1)\right) \cdot n \log n}, \quad \text{if } P_{succ}(\mathrm{ENUM}) = 1 - o(1).$$

For $P_{succ}(\mathrm{ENUM})$ arbitrary, the above quantity is a lower bound for $\Upsilon(\mathit{BDD})$.

Proof. We start with the case $P_{\mathrm{succ}}(\mathrm{ENUM}) = 1 - o(1)$. Since the running time $T(\mathrm{ENUM})$ drops if $T(\mathsf{BDD})$ is increased, the total running time $T(\mathsf{BDD}) = T(\mathtt{BKZ}) + T(\mathrm{ENUM})$ is minimized (up to a factor of at most 2) when the two phases of the attack are balanced: $T(\mathtt{BKZ}) = T(\mathrm{ENUM})$. On a logarithmic scale, using the result of Thms. 3 and 5, this condition is equivalent to (ignoring the $o(1)$ term):

$$\tfrac{1}{2}\left(\tfrac{m}{2\beta} - c_q + \tfrac{n}{m} c_q + c_s\right)^2 \beta \log \beta = c_{\mathtt{BKZ}} \beta \log \beta, \tag{9}$$

from where we easily derive $\beta = \frac{1}{2} \frac{m}{\sqrt{2c_{\mathtt{BKZ}}}+(1-n/m)c_q-c_s} = \Theta(m)$.

The obtained expression for β is minimized when we take $m = n \cdot \frac{2c_q}{\sqrt{2c_{\mathtt{BKZ}}}+c_q-c_s}$ samples. For such a choice, the first statement of the theorem follows.

If $P_{\mathrm{succ}}(\mathrm{ENUM})$ is arbitrary, the running time to success probability ratio satisfies $\Upsilon(\mathsf{BDD}) = \frac{T(\mathtt{BKZ})+T(\mathrm{ENUM})}{P_{\mathrm{succ}}(\mathrm{ENUM})} = \frac{T(\mathtt{BKZ})}{P_{\mathrm{succ}}(\mathrm{ENUM})} + \Upsilon(\mathrm{ENUM}) \leq T(\mathtt{BKZ}) + \Upsilon(\mathrm{ENUM})$ and we really just bounded $T(\mathtt{BKZ}) + \Upsilon(\mathrm{ENUM})$.

Now we consider the case when the lattice-reduction has complexity $T_{\mathtt{BKZ}} = 2^{\Theta(\beta)}$. Thm. 5 shows that for any generalized pruning strategy, the trade-off $\Upsilon(\mathrm{ENUM})$ is lower-bounded by $2^{\Theta(\beta \log \beta)}$ when run on a $\beta = \Theta(n)$-reduced basis as long as $\|\mathbf{b}_1\|$ is larger than s by a polynomial factor. Conversely, if $\|\mathbf{b}_1\|$ is smaller than s by any polynomial factor, enumeration becomes very easy: we achieve success probability $1 - o(1)$ in polynomial time by Babai's algorithm.

Thus, if we have a fast *single-exponential* reduction, the asymptotically optimal trade-off is to reduce the basis to the transition point where the cost of enumeration

switches from super-exponential to polynomial. We might not know the behavior of enumeration algorithms at exactly the transition point, but increasing β even slightly will not affect reduction in leading order and make enumeration truly polynomial.

Hence, the complexity of the whole attack boils down to the complexity of the lattice-reduction step, as the enumeration is comparatively cheap. What remains is to determine for which values of β we should run the reduction.

Theorem 7. *With a β-reduced basis reduction running in single-exponential time $2^{c_{BKZ}\beta}$, the complexity of solving the LWE problem with parameters $(n, q = \mathcal{O}(c_q), s = \mathcal{O}(c_s))$ via BDD using the optimal choice of $m = n \cdot \left(\frac{2c_q}{c_q - c_s} + o(1)\right)$ samples and block size $\beta = n \cdot \left(\frac{2c_q}{(c_q-c_s)^2} + o(1)\right)$ is*

$$T(\mathsf{BDD}) = 2^{\left(c_{BKZ} \cdot \frac{2c_q}{(c_q-c_s)^2} + o(1)\right) \cdot n}, \quad \text{with } P_{succ}(\mathsf{BDD}) = 1 - o(1).$$

Proof. To guarantee a constant success probability for a polynomial-time enumeration step, we set (cf. Thm. 2) $\frac{m}{2\beta} - (1 - n/m)c_q + c_s < 0$, yielding

$$\beta > \tfrac{1}{2} \frac{m}{(1-n/m)c_q - c_s}.$$

This value attains its minimum for $m = n \cdot \frac{2c_q}{c_q - c_s}$, from where we get $\beta > n \cdot \frac{2c_q}{(c_q-c_s)^2}$.

5 Embedding

A standard technique to convert a CVP instance $(\mathscr{L}(\mathbf{B}), \mathbf{t})$ to an SVP instance is due to Kannan ([26]): we consider a higher-dimensional lattice $\mathscr{L}_{\text{Embed}}(\mathbf{B}, \mathbf{t})$ spanned by $\{(\mathscr{L}(\mathbf{B}) \times \{0\}), (\mathbf{t}, \tau)\}$, where the so-called embedding factor τ should be large enough (half of a shortest vector of $\mathscr{L}(\mathbf{B})$ in the worst-case). Once a shortest vector in $\mathscr{L}_{\text{Embed}}(\mathbf{B})$ is of the form $(\mathbf{v}, e)$ and $e \neq 0$, then $\mathbf{v}$ is a solution to the original CVP instance.

Similarly, a BDD instance can be reduced to a so-called *γ-unique-SVP* problem, where we have the promise that the so-called *gap* $\frac{\lambda_2}{\lambda_1}$ is at least γ. Here, λ_1 resp. λ_2 are the first resp. second minima of the lattice. In the LWE case, a bound on the length of the error-vector allows us to estimate λ_1 and λ_2 in $\mathscr{L}_{\text{Embed}}(\mathbf{B}, \mathbf{t})$ as follows: $\lambda_1^2 = \|\mathbf{e}\|^2 + \tau^2$ and $\lambda_2 = \lambda_1(\mathscr{L}(\mathbf{B}))$. This gives us a bound γ on $\frac{\lambda_2}{\lambda_1}$ and thus it is enough to approximate the shortest vector by a factor of γ. We refer the reader to [35] for a reduction between unique-SVP and BDD.

The obvious tool to solve the *γ-unique-SVP* problem is lattice-basis reduction. Eq. (2) indicates that a β-BKZ-reduced basis of an m-dimensional lattice achieves an approximation factor of $\approx \beta^{m/(2\beta)}$ to a shortest vector. Thus, knowing the gap for an LWE lattice, we can estimate the required block-size β as follows (an analogous result, but in terms of the Hermite-root factor δ, is presented in [6]; also our choice of m is different):

Theorem 8. *With a β-BKZ basis reduction running in time $2^{c_{BKZ} \cdot f(\beta)}$, with $f(\beta) = \beta$ or $f(\beta) = \beta \log \beta$, the complexity of solving the LWE problem with parameters $(n, q = \mathcal{O}(n^{c_q}), s = \mathcal{O}(n^{c_s}))$ via embedding is*

$$T(\text{EMBED}) = 2^{\left(c_{BKZ} \cdot \frac{2c_q}{(c_q-c_s)^2} + o(1)\right) f(n)} \quad \text{with } m = \frac{(2+o(1))c_q}{c_q - c_s} \cdot n \text{ samples.}$$

Proof. Let us first estimate the gap for $\mathscr{L}_{\text{Embed}}(\mathbf{B})$, spanned by $\{(\mathscr{L}(\mathbf{B}) \times \{0\}), (\mathbf{t}, \|\mathbf{e}\|)\}$, where $\mathbf{B}$ is a basis for an m-dimensional lattice and $\|\mathbf{e}\| = \Theta(s\sqrt{m})$. Then,

$$\lambda_1(\mathscr{L}_{\text{Embed}}(\mathbf{B})) = \Theta(s\sqrt{m}), \quad \text{and} \quad \lambda_2(\mathscr{L}_{\text{Embed}}(\mathbf{B})) = \lambda_1(\mathscr{L}(\mathbf{B})) \leq \sqrt{m}q^{1-n/m},$$

by Minkowski's bound (more precisely, $\lambda_1(\mathscr{L}(\mathbf{B})) \leq \min\{q, \sqrt{m}q^{1-n/m}\}$, since we have a q-ary lattice, but for our choice of m, Minkowski's bound is always smaller). The value for β that achieves a sufficient approximation for given LWE-gap satisfies

$$\beta^{m/(2\beta)} = \frac{\lambda_2(\mathscr{L}_{\text{Embed}}(\mathbf{B}))}{\lambda_1((\mathscr{L}_{\text{Embed}}(\mathbf{B})))} = \Theta\big(\frac{q^{1-n/m}}{s}\big),$$

if we assume that Minkowski's bound $\lambda_1(\mathscr{L}(\mathbf{B})) \leq \sqrt{m}q^{1-n/m}$ holds with equality (the so-called Minkowski Heuristic). We obtain $\beta = \frac{1}{2}\frac{m}{(1-n/m)c_q - c_s} + o(m)$. Its global minimum is at $m = \frac{(2+o(1))c_q}{c_q - c_s} \cdot n$, leading to $\beta = \frac{(2+o(1))c_q}{(c_q-c_s)^2} \cdot n$.

Thus the embedding technique achieves the same constant in the exponent as the single-exponential reduction + polynomial-time enumeration algorithm (cf. Thm. 7). From the algorithmic point of view, both methods are nearly equivalent: performing polynomial-time enumeration (Babai) on a reduced basis can be seen as embedding the target vector in such a reduced basis and then size-reducing it.

6 Lattice Reduction on the Kernel

Another approach to solve the LWE problem is by lattice reduction on the (scaled) dual lattice. This attack already appears in [38] (but is analyzed in terms of the root-Hermite factor δ and not in our setting). An analysis matching our asymptotic approach is given, e.g. in [27, Full Version], which we briefly recall.

The main difference to the lattice attacks from section Sect. 4 is that we directly solve an instance of the (approximate) shortest vector problem SVP (rather than a promise version of CVP) and that this attack is more naturally viewed as an attack against the *decision* version of LWE.

More precisely, given an LWE instance $(\mathbf{A}, \mathbf{t} = \mathbf{A}^{\mathsf{t}}\mathbf{s} + \mathbf{e} \bmod q)$ with $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ we consider the q-ary lattice

$$\Lambda_q^{\perp}(\mathbf{A}^{\mathsf{t}}) = \big\{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} \bmod q = 0\big\} \ .$$

Its dimension is m and, provided $\mathbf{A}$ has full rank, its determinant is $\det\Lambda_q^{\perp}(\mathbf{A}^{\mathsf{t}}) = q^n$. We use BKZ to reduce this lattice and find a short non-zero vector $\mathbf{v} \in \Lambda_q^{\perp}(\mathbf{A}^{\mathsf{t}})$. Given such a vector $\mathbf{v}$, we can compute $w := \langle \mathbf{v}, \mathbf{t} \rangle \bmod q = \mathbf{v}^{\mathsf{t}}(\mathbf{A}^{\mathsf{t}}\mathbf{s} + \mathbf{e}) \bmod q = \mathbf{v}^{\mathsf{t}}\mathbf{e} \bmod q = \langle \mathbf{v}, \mathbf{e} \rangle \bmod q$. If $\mathbf{e}$ is uniform, the resulting $\langle \mathbf{v}, \mathbf{t} \rangle \bmod q$ is uniform, whereas if $\mathbf{e}$ is Gaussian with standard deviation s, then $\langle \mathbf{v}, \mathbf{t} \rangle = \sum_i v_i e_i$ is Gaussian again (if we pretend for a moment that the discrete Gaussians e_i were continuous) with standard deviation $s \cdot \|\mathbf{v}\|$. The statistical distance of $w \bmod q$ to a uniform random variable $\bmod q$ is then given by $\alpha = 2^{-\mathscr{O}\left(\frac{s\|\mathbf{v}\|}{q}\right)^2}$. This still holds true for our discrete Gaussian setting and

efficient distinguishers reaching advantage $\Theta(\alpha)$ exist[7]. This means that we should aim for $\|\mathbf{v}\| = \mathscr{O}(n^{1/2+c_q-c_s-\varepsilon})$ for any $\varepsilon > 0$ to obtain sub-exponential advantage. By Eq. (2), we have $\|\mathbf{v}\| = \mathscr{O}(\beta^{\frac{m}{2\beta}} q^{\frac{n}{m}})$. Optimization and letting $\varepsilon \to 0$ leads to

$$m = \Big(\frac{2c_q}{1/2+c_q-c_s} + o(1)\Big) \cdot n, \qquad \beta = \Big(\frac{2c_q}{(1/2+c_q-c_s)^2} + o(1)\Big) n \ .$$

To solve the *search* problem with constant probability, we can use the generic reduction from [43] or use a more efficient Fourier-based approach [7], losing a factor of $\mathrm{poly}(n) \cdot \alpha^{-2}$ in the running time, which does not affect the asymptotics for $\varepsilon > 0$. The memory complexity of the reduction can be made polynomial. Note that the resulting running time is better than those from Sect. 4 due to the additional $\frac{1}{2}$-term in the denominator. However, solving the search problem entails repeating the algorithm against the decision problem $\mathrm{poly}(n)\alpha^{-2}$ many times with independent inputs. If we only have polynomially many inputs, as is typically the case, we can either lower the bound on $\|\mathbf{v}\|$ or use sample amplification as detailed in Sect. 7. This directly leads to the following running times, where we interpolate between these cases using $\|\mathbf{v}\| = \mathscr{O}(n^{c_q-c_s+\gamma/2})$. Note that amplification leads to the same results as lowering $\|\mathbf{v}\|$ for $\gamma \in \{0,1\}$. For $0 < \gamma < 1$, we conjecture that this holds as well (cf. Rmk. 12).

Theorem 9. *With a β-BKZ reduction running in time $2^{c_{BKZ} f(\beta)}$, with $f(\beta) = \beta$ or $f(\beta) = \beta \log \beta$, the complexity of solving the LWE problem with parameters $(n, q = \mathscr{O}(n^{c_q}), s = \mathscr{O}(n^{c_s}))$ via lattice reduction on the dual is*

$$T(\mathtt{DUAL}) = 2^{\left(c_{BKZ} \cdot \frac{2c_q}{(c_q-c_s)^2} + o(1)\right) \cdot f(n)}$$

using $m = \Omega(n \log n)$ many samples or

$$T(\mathtt{DUAL}) = 2^{\left(c_{BKZ} \cdot \frac{2c_q}{(\gamma/2+c_q-c_s)^2} + o(1)\right) \cdot f(n)}$$

using $m = 2^{\mathscr{O}(n^\gamma)}$ samples for $0 < \gamma \leq 1$. The success probability is $1 - o(1)$. The memory complexity is dominated by the β-BKZ reduction where $\beta = \Theta(n)$.

Note that for $\gamma > 0$, we need not store all the samples simultaneously, but rather need to query m samples.

7 BKW

7.1 Original BKW

The BKW algorithm [13] is an algorithm designed originally for LPN (i.e. LWE with $q = 2$ and Bernoulli distributed noise) and later generalized to LWE in [4]. It consists of two main phases. The BKW algorithm uses a very large initial number m of samples of the form $(\mathbf{a}, \langle \mathbf{a}_i, \mathbf{s} \rangle + e)$, where $m = 2^{\Theta(n/\log n)}$ in the case of LPN and $m = 2^{\Theta(n)}$ for LWE with our parameters. In the first phase, it adds/subtracts pairs of such samples to construct new samples of the form $(\mathbf{a}', \langle \mathbf{a}', \mathbf{s} \rangle + e')$ where the pairs are selected via a

collision finding algorithm such that several chosen coordinates of $\mathbf{a}'$ are 0. Iterating this process k times, we obtain a large number of final samples $(\mathbf{a}'', \langle \mathbf{a}'', \mathbf{s}\rangle + e'')$ where only a small number of coordinates of $\mathbf{a}''$ are possibly non-zero. In the second phase, it determines $\mathbf{s}$ with high probability on those coordinates (e.g. by exhaustive search over the possible values of $\mathbf{s}$ on those coordinates). The rest of the coordinates of $\mathbf{s}$ can be obtained in the same fashion. We refer to [13] resp. [4] for more details. Note that the BKW algorithm for LWE can be viewed as a way to find short vectors in the kernel lattice $\Lambda_q^\perp(\mathbf{A}^\mathsf{t})$ in a combinatorial way rather than via lattice reduction as in Sect. 6: if we combine samples until actually *all* coordinates of $\mathbf{a}''$ are 0, the selection of original samples used to create $\mathbf{a}''$ corresponds to a short $\mathbf{v} \in \Lambda_q^\perp(\mathbf{A}^\mathsf{t})$ with entries from $\{0, \pm 1\}$ such that $\|\mathbf{v}\|^2 = 2^k$.[8] In particular, the same considerations as in Sect. 6 on the allowed length of $\|\mathbf{v}\|$ apply. Not reducing all coordinates to 0 can be viewed as a way to make the search-to-decision reduction more efficient and can also be done in Sect. 6. In fact, there are various improvements for BKW (e.g. [5, 7]) that improve the second phase. While relevant in practice, these do not change the asymptotics, since the second phase is not relevant for the asymptotics.

Since we need an exponential number of samples anyway, both initially and at every intermediate step, it is not a problem to produce an exponentially large amount of final samples and we should set $\|\mathbf{v}\| = \mathcal{O}(n^{1/2 + c_q - c_s})$, so the noise in the final samples has standard deviation $\mathcal{O}(\sqrt{n}q)$.

For the LWE setting with parameters $q = \mathcal{O}(n^{c_q}), s = \mathcal{O}(n^{c_s})$, by the result of [27], the complexity of the BKW algorithm is single-exponential, namely

$$T(\mathtt{BKW}) = 2^{\left(\frac{c_q}{2\cdot(1/2 + c_q - c_s)} + o(1)\right)\cdot n} \tag{10}$$

(in time, memory and LWE samples). The success probability is close to 1. Note that the original analysis of [4] lacks the $1/2$-term, because the authors chose a suboptimal $\|\mathbf{v}\| = n^{c_q - c_s}$.

7.2 BKW2

Recently, [27] and [22] independently proposed a modification to the first phase of BKW that changes the asymptotics. We will call the resulting algorithm BKW2 for comparison. At the expense of $n + o(n)$ samples, we can achieve that the entries of the secret follow the same distribution as the noise and are thus small[9]. In the original BKW, most/all coordinates of $\mathbf{a}''$ were 0 in the final samples $(\mathbf{a}'', \langle \mathbf{a}'', \mathbf{s}\rangle + e'')$. In BKW2 this is relaxed to only requiring that $\|\mathbf{a}''\|$ be small. The term $\langle \mathbf{a}'', \mathbf{s}\rangle$ can then be treated as another noise term (which is chosen to have about the same order of magnitude as e''). For our setting with $q = \mathcal{O}(n^{c_q}), s = \mathcal{O}(n^{c_s})$, the analysis of [27] yields

$$T(\mathtt{BKW2}) = 2^{\left(\frac{1}{c_q} + 2\ln(\frac{c_q}{c_s}) + o(1)\right)^{-1}\cdot n} \tag{11}$$

[8] This depends on to what extent we allow reusing the same initial sample in several combinations, as allowed by some variants [31] of BKW at the expense of a heuristic analysis. These modifications have no impact on the asymptotics.

(in time, memory and LWE samples). The success probability is close to 1. Note that this assumes that the coefficients of the secret follow the same distribution as the noise for simplicity and easier comparison. If the distribution of $\mathbf{s}$ is such that the coefficients of the secret are even smaller than those of the noise, the BKW2 algorithm becomes even better than that.

7.3 Amplification

In most cases, an LWE-based scheme produces only $m = \text{poly}(n)$ number of LWE samples (where the polynomial bound can be as small as $m = \Theta(n)$). The BKW algorithm, however, requires exponentially many of them. Thus, we would like to be able to produce many new random-looking LWE samples out of m given ones ("sample amplification") and run BKW on those new samples.

For the LPN (i.e. $q = 2$) case, an analysis of such amplification was made by Lyubashevsky in [34]. One generates exponentially many samples given only $m = n^{1+\varepsilon}$ initial ones by xor-ing a random sparse subset of the m given samples.

To ensure that we can use the LPN samples of the form $(\mathbf{a}', c') \in \mathbb{Z}_2^n \times \mathbb{Z}_2$ generated by this amplification process in BKW, [34] verifies that the following conditions are satisfied: up to a statistically small deviation D,

1. $\mathbf{a}'$ is uniformly distributed and independent of the initial m samples.
2. the new noise in c' is independent from $\mathbf{a}'$, even conditioned on the initial samples.
3. the BKW algorithm can work with the new noise distribution in c'.

To argue about the first two condition, [34] invokes the Leftover Hash Lemma. As for the third condition, the noise-rate in the new output samples is much larger than the noise of the input samples, which in turn slows down the BKW algorithm from $2^{\mathcal{O}(\frac{n}{\log n})}$ to $2^{\mathcal{O}(\frac{n}{\log\log n})}$. Note that the statistically small deviation D not only has to be negligibly small, but rather has to be small compared to the inverse of number of samples consumed by BKW.

Similar techniques are known for LWE (e.g. [9, 19], as mentioned in [4]), requiring $n \log n$ many samples to simulate a wider Gaussian within negligibly small statistical distance. Unfortunately, negligibly small statistical distance is not quite enough, so we need a more detailed analysis. The same technique as in [34] has been used for LWE in [7]. However, the authors do not provide a complete argumentation of why the recycled LWE samples meet the aforementioned requirements, in particular, [7, Theorem 18] argues only about the first one. Actually, since the first phase of BKW is collision-finding, a non-uniform distribution of $\mathbf{a}'$ would *improve* the running time of BKW, so we do not even need the first requirement. The crucial point really is showing the second requirement.

The recent [27] adapts Döttling's reduction (which is a computational argument, avoiding the statistical Leftover Hash Lemma)[15] from the LPN to the LWE case to prove this. However, in turning this into a statistical argument in[27, Corollary 6 in Full Version], there is a flaw in the proof (The hypothesis of the Corollary should read $\omega(\log k)$ rather than $\omega(1)$), making the proof inapplicable for $c_s > 0$.

Regarding the third requirement, note that the BKW algorithm (or BKW2 or the approach from Sect. 6) expects the error to follow a discrete Gaussian. The error e'' in the

final samples after the first phase of BKW will be an appropriate sum of discrete Gaussians. If we change the input to BKW to a sum of discrete Gaussians, e'' is still a sum of discrete Gaussians, with more summands and the individual summands following a narrower distribution. This will not affect the asymptotic behavior of BKW. A refined analysis of this effect of using a *discrete* Gaussian (so sums of independent Gaussians are no longer Gaussian) was performed in [7], using the logarithmic bias to measure the width of the involved distributions and also using it to distinguish (appearing more explicitly in [27]). More precisely, for a noise distribution χ on $\mathbb{Z}_q$ (q odd prime) with $\chi(-x) = \chi(x)$, consider $\mathbb{LB}(\chi) := -q^2 \log \widehat{\chi}(1)$ for $\widehat{\chi}(u) = \sum_x \chi(x) e^{\frac{2\pi i x u}{q}}$. The logarithmic bias $\mathbb{LB}$ is additive for sums of independent variables (due to the convolution theorem) and is the only relevant quantity we care for in our noise distributions. We can distinguish from uniform in sub-exponential time as long as $\sqrt{\mathbb{LB}(\chi)} = \mathcal{O}(qn^{1/2-\varepsilon})$ for $\varepsilon > 0$. As we have $\mathbb{V}(\chi) = \Theta(1) \cdot \mathbb{LB}(\chi)$ for all relevant noise distributions we encounter, we will consider the more familiar notions of standard deviation/variance instead in our proofs.

In the theorem below we refine the analysis of LWE sample amplification and show how the noise-rate growth affects the running time of the BKW algorithms. Our sample amplification works as follows: Given m initial LWE samples $(\mathbf{A}, \mathbf{t} = \mathbf{A}^{\mathsf{t}}\mathbf{s} + \mathbf{e} \bmod q)$, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we produce new samples by taking $\mathbf{x}$ from a discrete Gaussian on $\mathbb{Z}^m$ of some width ω and output

$$\Big(\mathbf{A}\mathbf{x} \bmod q, \langle \mathbf{t}, \mathbf{x} \rangle \bmod q\Big) = \Big(\mathbf{A}\mathbf{x} \bmod q, \langle \mathbf{A}\mathbf{x}, \mathbf{s} \rangle + \langle \mathbf{e}, \mathbf{x} \rangle \bmod q\Big).$$

Lemma 10. *Consider m initial LWE samples $(\boldsymbol{A}, \boldsymbol{t} = \boldsymbol{A}^{\mathsf{t}}\boldsymbol{s} + \boldsymbol{e} \bmod q)$ with parameters $q = \Theta(n^{c_q})$ and $s = \Theta(n^{c_s})$ for $c_q, c_s = \Theta(1)$ with $c_q > c_s + \frac{1}{2}$. Let $\boldsymbol{x}$ be a Gaussian on $\mathbb{Z}^m$ for some parameter $\omega = \Omega(1)$ and set $e' = \langle \boldsymbol{e}, \boldsymbol{x} \rangle$. We treat $\boldsymbol{A}, \boldsymbol{t}, \boldsymbol{s}, \boldsymbol{e}, \boldsymbol{x}, e'$ as random variables. For any fixed $\widetilde{\boldsymbol{e}} \in \mathbb{Z}^m$, we consider the statistical distance*

$$D_{\widetilde{\boldsymbol{e}}} := \mathrm{SD}((\boldsymbol{A}, \boldsymbol{s}, \boldsymbol{e}, \boldsymbol{A}\boldsymbol{x} \bmod q, e' \bmod q) \mid (\boldsymbol{e} = \widetilde{\boldsymbol{e}}); (\boldsymbol{A}, \boldsymbol{s}, \boldsymbol{e}, \boldsymbol{u}, e' \bmod q) \mid (\boldsymbol{e} = \widetilde{\boldsymbol{e}})),$$

where $\boldsymbol{u} \in \mathbb{Z}_q^m$ is uniform (and independent of anything else). Then the following holds:

1. *$e' \mid (\boldsymbol{e} = \widetilde{\boldsymbol{e}})$ follows a distribution with variance $\Theta(\omega^2 \|\widetilde{\boldsymbol{e}}\|_2^2)$.*
2. *With probability $1 - o(1)$, we have $0 < \|\boldsymbol{e}\|_\infty < s \log m$ and $\|\boldsymbol{e}\|_2 < 10\sqrt{m}s$.*
3. *For any $\widetilde{\boldsymbol{e}}$ s.t. $0 < \|\widetilde{\boldsymbol{e}}\|_\infty < s \log m$, we have that $D_{\widetilde{\boldsymbol{e}}} < 2^{-\Omega(n \log n)}$, provided that either*
   - *$m = \Theta(n \log n)$ and ω a sufficiently large constant or*
   - *$m = c_m \cdot n, \omega = \Theta(n^{c_\omega})$ with $c_m, c_\omega = \Theta(1)$ and $\frac{c_q}{c_m} < c_\omega < c_q - \frac{1}{2}$.*

Note that we consider the entries of $\boldsymbol{e}, e', \boldsymbol{x}$ as elements from $\mathbb{Z}$ and not from $\mathbb{Z}_q$.

The lemma, whose proof will be given below, readily implies the following theorem:

Theorem 11. *The BKW algorithms (BKW and BKW2), given $m = \Theta(n \log n)$ LWE samples with parameters $(n, q = n^{c_q}, s = n^{c_s})$ for constants $c_q > c_s + \frac{1}{2}$, solve the search-LWE problem in time and memory*

$$T(\mathit{BKW}_{\mathit{Amplfied}}) = 2^{\left(\frac{c_q}{2c_q - 2c_s} + o(1)\right) \cdot n} \quad \textit{resp.}$$

$$T(\mathit{BKW2}_{\mathit{Amplfied}}) = 2^{\left(\frac{1}{2\ln(c_q/c_s)} + o(1)\right) \cdot n}$$

with success probability $1-o(1)$*. Given only* $m=(c_m+o(1))\cdot n$ *LWE samples with* $c_m=\Theta(1)$ *such that* $c_q>c_s+\frac{1}{2}+\frac{c_q}{c_m}$ *and* $\frac{c_q}{c_m}<c_q-\frac{1}{2}$ *for BKW resp.* $c_q>c_s+\frac{1}{2}+\frac{c_q}{c_m-1}$ *and* $\frac{c_q}{c_m-1}<c_q-\frac{1}{2}$ *for BKW2, the BKW algorithms solve the search-LWE problem with success probability* $1-o(1)$ *in time and memory*

$$T(BKW_{Amplfied})=2^{\left(\frac{c_q}{2c_q-2c_s-2c_q/c_m}+o(1)\right)\cdot n},$$
$$T(BKW2_{Amplfied})=2^{\left(2\ln(\frac{c_q-c_q/(c_m-1)}{c_s})+o(1)\right)^{-1}\cdot n}.$$

Proof (of Thm. 11). We use our amplification strategy from above and run BKW on the simulated samples. With probability $1-o(1)$, the noise in the initial m samples is small enough so we can apply Lemma 10. Since BKW only requires an exponential number of samples and the statistical distance is superexponentially small, BKW works with those amplified samples. Note for BKW2 that we lose $n+o(n)$ samples to change the distribution of the coefficients of the secret to that of the noise prior to amplification, hence the c_m-1-term. Then, sample amplification increases the level of noise, but not the size of the secret. Due to that, the running time of BKW2 is better that what we would obtain by plugging in $c_s+\frac{1}{2}$ resp. $c_s+\frac{1}{2}+\frac{c_q}{c_m-1}$ into Eq. (11). We rather used [27, Thm. 4/Thm. 5].

Proof (of Lemma 10). The variance $\mathbb{V}[e' \mid (\mathbf{e}=\widetilde{\mathbf{e}})]$ of $e' \mid (\mathbf{e}=\widetilde{\mathbf{e}})$ is given by

$$\mathbb{V}\Big[\sum_{i=1}^{m}\mathbf{x}_i\widetilde{\mathbf{e}}_i\Big]=\sum_{i=1}^{m}\widetilde{e}_i^2\,\mathbb{V}[\mathbf{x}_i]=\|\widetilde{\mathbf{e}}\|^2\cdot\Theta(\omega^2).$$

Statement 2 follows from Eq. (1) and a union bound. For statement 3, we write $D_{\widetilde{\mathbf{e}}}$ as an expected value of statistical distances between conditional distributions: since $\mathrm{SD}((A,B);(A,C))=\mathbb{E}_A[\mathrm{SD}(B\mid A;C\mid A)]$ for any A,B,C, we have

$$D_{\widetilde{\mathbf{e}}}=\mathop{\mathbb{E}}_{e'\mid(\mathbf{e}=\widetilde{\mathbf{e}})}\left[\mathrm{SD}\left((\mathbf{A},\mathbf{A}\mathbf{x} \bmod q)\mid(\mathbf{e}=\widetilde{\mathbf{e}},e');(\mathbf{A},\mathbf{u})\right)\right]. \tag{12}$$

For fixed $\widetilde{\mathbf{e}}$ and $\widetilde{e}'$ with $0<\|\widetilde{\mathbf{e}}\|_\infty<s\log m$, let us define sets

$$X_{\widetilde{\mathbf{e}},\widetilde{e}'}=\{\widetilde{\mathbf{x}}\mid\langle\widetilde{\mathbf{x}},\widetilde{\mathbf{e}}\rangle=\widetilde{e}'\}\subset\mathbb{Z}^m \quad\text{and}\quad X^q_{\widetilde{\mathbf{e}},\widetilde{e}'}=\{\widetilde{\mathbf{x}} \bmod q\mid\langle\widetilde{\mathbf{x}},\widetilde{\mathbf{e}}\rangle=\widetilde{e}'\ (\text{in }\mathbb{Z})\}\subset\mathbb{Z}_q^m$$
$$\text{and the lattice } \mathscr{L}_{\widetilde{\mathbf{e}}}=\{\widetilde{\mathbf{x}}\mid\langle\widetilde{\mathbf{x}},\widetilde{\mathbf{e}}\rangle=0\}\subset\mathbb{Z}^m.$$

Since $\widetilde{\mathbf{e}}\neq 0$, $\mathscr{L}_{\widetilde{\mathbf{e}}}$ has dimension $m-1$. Its determinant is $\det(\mathscr{L}_{\widetilde{\mathbf{e}}})=\frac{\|\widetilde{\mathbf{e}}\|}{\gcd(\widetilde{\mathbf{e}})}$. Note that $X_{\widetilde{\mathbf{e}},\widetilde{e}'}$ is empty if $\gcd(\widetilde{\mathbf{e}})$ does not divide $\widetilde{e}'$. We ignore this case, as is does not contribute to Eq. (12) anyway. So $X_{\widetilde{\mathbf{e}},\widetilde{e}'}$ is a shifted copy of $\mathscr{L}_{\widetilde{\mathbf{e}}}$. The set $X_{\widetilde{\mathbf{e}},\widetilde{e}'}$ carries a probability distribution with probability density proportional to

$$\rho_\omega(\widetilde{\mathbf{x}})=\exp(-\pi\|\widetilde{\mathbf{x}}\|^2/\omega^2),$$

which induces a probability distribution on $X^q_{\widetilde{\mathbf{e}},\widetilde{e}'}$ by reducing $\bmod q$. For fixed $\widetilde{\mathbf{e}},\widetilde{e}'$, the family of hash functions

$$H^{\widetilde{\mathbf{e}},\widetilde{e}'}_{\mathbf{A}}:\ X^q_{\widetilde{\mathbf{e}},\widetilde{e}'}\to\mathbb{Z}_q,\widetilde{\mathbf{x}}\mapsto\mathbf{A}\widetilde{\mathbf{x}}$$

(with hash key $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$) is universal. Hence, by the Leftover Hash Lemma, we obtain

$$\mathrm{SD}\left((\mathbf{A}, \mathbf{Ax} \bmod q) \mid (\mathbf{e} = \widetilde{\mathbf{e}}, e' = \widetilde{e}'); (\mathbf{A}, \mathbf{u})\right) \leq 2^{-\frac{1}{2}(H_\infty(X^q_{\widetilde{\mathbf{e}},\widetilde{e}'}) - c_q n \log n)},$$

where $H_\infty(X^q_{\widetilde{\mathbf{e}},\widetilde{e}'})$ denotes the min-entropy of $X^q_{\widetilde{\mathbf{e}},\widetilde{e}'}$ wrt. the distribution from above. So it suffices to show that $H_\infty(X^q_{\widetilde{\mathbf{e}},\widetilde{e}'}) - c_q n \log n = \Omega(n \log n)$. Let $\mathbf{x}^*$ be a vector in $X_{\widetilde{\mathbf{e}},\widetilde{e}'}$ with minimal $\|.\|_2$-norm. By construction, $X_{\widetilde{\mathbf{e}},\widetilde{e}'} = \mathbf{x}^* + \mathscr{L}_{\widetilde{\mathbf{e}}}$ and we can translate everything from $X_{\widetilde{\mathbf{e}},\widetilde{e}'}$ to $\mathscr{L}_{\widetilde{\mathbf{e}}}$. Under this translation by $-\mathbf{x}^*$, the distribution on $X_{\widetilde{\mathbf{e}},\widetilde{e}'}$ induces the probability distribution $\rho_{\omega,\mathbf{u}}$ on $\mathscr{L}_{\widetilde{\mathbf{e}}}$, which is a discrete Gaussian with parameter ω, centered around some $\mathbf{u} \in \mathrm{Span}\,\mathscr{L}_{\widetilde{\mathbf{e}}}$. We claim that $\|\mathbf{u}\|_2 = o(q)$:

by minimality of $\mathbf{x}^*$, the origin is a closest lattice point to $\mathbf{u}$ in $\mathscr{L}_{\widetilde{\mathbf{e}}}$, so $\|\mathbf{u}\|$ is at most the covering radius of $\mathscr{L}_{\widetilde{\mathbf{e}}}$ (cf. [36, Def. 7.10]). To estimate the covering radius, note that for $i < j$, we have vectors of the form $\mathbf{v}_{ij} = (0, \ldots, 0, \widetilde{\mathbf{e}}_j, 0, \ldots, 0, -\widetilde{\mathbf{e}}_i, 0, \ldots) \in \mathscr{L}_{\widetilde{\mathbf{e}}}$ and these vectors span an $m-1$-dimensional space. It follows via [36, Thm. 7.9] that $\|\mathbf{u}\|_2 \leq \sqrt{m-1} \cdot \sqrt{2}\, \|\widetilde{\mathbf{e}}\|_\infty = o(q)$.

To estimate the min-entropy of $X^q_{\widetilde{\mathbf{e}},\widetilde{e}'}$, we need to bound

$$\max_{\widetilde{\mathbf{x}} \in \mathscr{L}_{\widetilde{\mathbf{e}}}} \frac{\rho_\omega(\widetilde{\mathbf{x}} - \mathbf{u} + q\mathscr{L}_{\widetilde{\mathbf{e}}})}{\rho_\omega(-\mathbf{u} + \mathscr{L}_{\widetilde{\mathbf{e}}})}$$

from above. Note that the $q\mathscr{L}_{\widetilde{\mathbf{e}}} = (q\mathbb{Z}^m \cap \mathscr{L}_{\widetilde{\mathbf{e}}})$-term accounts for the difference between $X^q_{\widetilde{\mathbf{e}},\widetilde{e}'}$ and $X_{\widetilde{\mathbf{e}},\widetilde{e}'}$. To get rid of it, we use that

$$\rho_\omega\left(\widetilde{\mathbf{x}} - \mathbf{u} + q\mathscr{L}_{\widetilde{\mathbf{e}}}\right) = \rho_\omega\left(\left(\widetilde{\mathbf{x}} - \mathbf{u} + q\mathscr{L}_{\widetilde{\mathbf{e}}}\right) \cap \mathscr{B}(\tfrac{q}{2})\right) + \rho_\omega\left(\left(\widetilde{\mathbf{x}} - \mathbf{u} + q\mathscr{L}_{\widetilde{\mathbf{e}}}\right) \setminus \mathscr{B}(\tfrac{q}{2})\right), \quad (13)$$

where $\mathscr{B}(\frac{q}{2})$ denotes a Ball with radius $\frac{q}{2}$ in $\mathrm{Span}\,\mathscr{L}_{\widetilde{\mathbf{e}}}$ around the origin. We have

$$\rho_\omega\left(\left(\widetilde{\mathbf{x}} - \mathbf{u} + q\mathscr{L}_{\widetilde{\mathbf{e}}}\right) \cap \mathscr{B}(\tfrac{q}{2})\right) \leq \rho_\omega(-\mathbf{u}) = \exp(-\tfrac{\pi\|\mathbf{u}\|^2}{\omega^2}), \quad (14)$$

because a ball of radius $\frac{q}{2}$ can contain at most one point of $q\mathscr{L}_{\widetilde{\mathbf{e}}}$ and $\rho_\omega(\widetilde{\mathbf{x}} - \mathbf{u}) \leq \rho_\omega(-\mathbf{u})$ for any $\widetilde{\mathbf{x}} \in \mathscr{L}_{\widetilde{\mathbf{e}}}$ by minimality of $\mathbf{x}^*$. By [12, Lemma 1.5], setting $c := \frac{q}{2\omega\sqrt{m-1}}$, we get

$$\rho_\omega\left(\left(\widetilde{\mathbf{x}} - \mathbf{u} + q\mathscr{L}_{\widetilde{\mathbf{e}}}\right) \setminus \mathscr{B}(\tfrac{q}{2})\right) \leq 2\left(\sqrt{2\pi}ce^{-\pi c^2}\right)^{m-1} \rho_\omega(q\mathscr{L}_{\widetilde{\mathbf{e}}}) = 2^{\mathscr{O}(m \log m)} e^{-\frac{\pi q^2}{4\omega^2}} \rho_\omega(q\mathscr{L}_{\widetilde{\mathbf{e}}}).$$

Now, $\rho_\omega(q\mathscr{L}_{\widetilde{\mathbf{e}}}) = \rho_{\frac{\omega}{q}}(\mathscr{L}_{\widetilde{\mathbf{e}}}) \leq \rho_{\frac{\omega}{q}}(\mathbb{Z}^m) = \left(\rho_{\frac{\omega}{q}}(\mathbb{Z})\right)^m = 2^{\mathscr{O}(m)}$, since $\frac{\omega}{q} = \mathscr{O}(1)$. Further, since $m \log m = o(\frac{q^2}{\omega^2})$ and $\frac{\|\mathbf{u}\|^2}{\omega^2} = o(\frac{q^2}{\omega^2})$, it follows that

$$\rho_\omega\left(\left(\widetilde{\mathbf{x}} - \mathbf{u} + q\mathscr{L}_{\widetilde{\mathbf{e}}}\right) \setminus \mathscr{B}(\tfrac{q}{2})\right) = o(1) \cdot \exp(-\tfrac{\pi\|\mathbf{u}\|^2}{\omega^2}). \quad (15)$$

Combining Eqns. (13),(14),(15), we get

$$\max_{\widetilde{\mathbf{x}} \in \mathscr{L}_{\widetilde{\mathbf{e}}}} \tfrac{\rho_\omega(\widetilde{\mathbf{x}} - \mathbf{u} + q\mathscr{L}_{\widetilde{\mathbf{e}}})}{\rho_\omega(-\mathbf{u} + \mathscr{L}_{\widetilde{\mathbf{e}}})} \leq \mathscr{O}(1) \cdot f(\mathbf{u}) \quad \text{with} \quad f(\mathbf{u}) := \tfrac{\rho_\omega(-\mathbf{u})}{\rho_\omega(-\mathbf{u} + \mathscr{L}_{\widetilde{\mathbf{e}}})}. \quad (16)$$

We claim that $f(\mathbf{u})$ attains its maximal value for $\mathbf{u}=0$: indeed, computing the gradient $\operatorname{grad} f(\mathbf{u})$ of f gives $\operatorname{grad} f(\mathbf{u}) = \frac{-2\pi\rho_\omega(-\mathbf{u})}{\omega^2\rho_\omega(-\mathbf{u}+\mathscr{L}_{\widetilde{\mathbf{e}}})^2}\sum_{\widetilde{\mathbf{x}}\in\mathscr{L}_{\widetilde{\mathbf{e}}}}\widetilde{\mathbf{x}}\rho_\omega(\widetilde{\mathbf{x}}-\mathbf{u})$. We simplify this as

$$\begin{aligned}
\operatorname{grad} f(\mathbf{u}) &= \tfrac{-2\pi\rho_\omega(-\mathbf{u})}{\omega^2\rho_\omega(-\mathbf{u}+\mathscr{L}_{\widetilde{\mathbf{e}}})^2}\sum_{\widetilde{\mathbf{x}}\in\mathscr{L}_{\widetilde{\mathbf{e}}}}\widetilde{\mathbf{x}}\rho_\omega(\widetilde{\mathbf{x}}-\mathbf{u}) \\
&= \tfrac{-2\pi\rho_\omega(-\mathbf{u})}{\omega^2\rho_\omega(-\mathbf{u}+\mathscr{L}_{\widetilde{\mathbf{e}}})^2}\Big(\tfrac{1}{2}\sum_{\widetilde{\mathbf{x}}\in\mathscr{L}_{\widetilde{\mathbf{e}}}}\widetilde{\mathbf{x}}\rho_\omega(\widetilde{\mathbf{x}}-\mathbf{u}) + \tfrac{1}{2}\sum_{\widetilde{\mathbf{x}}\in\mathscr{L}_{\widetilde{\mathbf{e}}}}-\widetilde{\mathbf{x}}\rho_\omega(-\widetilde{\mathbf{x}}-\mathbf{u})\Big) \\
&= \tfrac{-2\pi\rho_\omega(-\mathbf{u})}{\omega^2\rho_\omega(-\mathbf{u}+\mathscr{L}_{\widetilde{\mathbf{e}}})^2}\sum_{\widetilde{\mathbf{x}}\in\mathscr{L}_{\widetilde{\mathbf{e}}}}\tfrac{1}{2}\widetilde{\mathbf{x}}e^{-\frac{\pi\widetilde{\mathbf{x}}^2+\pi\mathbf{u}^2}{\omega^2}}\Big(e^{\frac{2\pi\langle\widetilde{\mathbf{x}},\mathbf{u}\rangle}{\omega^2}} - e^{-\frac{2\pi\langle\widetilde{\mathbf{x}},\mathbf{u}\rangle}{\omega^2}}\Big) \\
&= \tfrac{-2\pi\rho_\omega(-\mathbf{u})^2}{\omega^2\rho_\omega(-\mathbf{u}+\mathscr{L}_{\widetilde{\mathbf{e}}})^2}\sum_{\widetilde{\mathbf{x}}\in\mathscr{L}_{\widetilde{\mathbf{e}}}}\widetilde{\mathbf{x}}\rho_\omega(\widetilde{\mathbf{x}})\sinh\Big(\frac{2\pi}{\omega^2}\langle\widetilde{\mathbf{x}},\mathbf{u}\rangle\Big).
\end{aligned}$$

Hence, $\operatorname{grad} f(0)=0$. Since $\langle\widetilde{\mathbf{x}},\mathbf{u}\rangle\cdot\sinh\big(\frac{2\pi}{\omega^2}\langle\widetilde{\mathbf{x}},\mathbf{u}\rangle\big)\geq 0$, we have $\langle\mathbf{u},\operatorname{grad} f(\mathbf{u})\rangle<0$ whenever $\mathbf{u}\neq 0$. Consequently, the only local extremum of f is at the origin, with $f(0)=\rho_\omega(\mathscr{L}_{\widetilde{\mathbf{e}}})^{-1}$, which must be the global maximum due to the asymptotic behavior of f. To bound $\rho_\omega(\mathscr{L}_{\widetilde{\mathbf{e}}})$, we write $\mathbb{Z}^m=\bigcup_{\widetilde{e}'\in\gcd(\widetilde{\mathbf{e}})\cdot\mathbb{Z}}X_{\widetilde{\mathbf{e}},\widetilde{e}'}$ as a union of translates of $\mathscr{L}_{\widetilde{\mathbf{e}}}$. The hyperplane spanned by $X_{\widetilde{\mathbf{e}},\widetilde{e}'}$ has distance $\frac{|\widetilde{e}'|}{\|\widetilde{\mathbf{e}}\|}$ to the origin, so we have

$$\begin{aligned}
\rho_\omega(\mathbb{Z}^m) &= \sum_{\widetilde{e}'}\rho_\omega(X_{\widetilde{\mathbf{e}},\widetilde{e}'}) = \sum_{\widetilde{e}'}\rho_\omega(\widetilde{e}'/\|\widetilde{\mathbf{e}}\|)\rho_\omega(\mathscr{L}_{\widetilde{\mathbf{e}}}+\mathbf{u}_{\widetilde{e}'}) \\
&\leq \sum_{\widetilde{e}'}\rho_\omega(\widetilde{e}'/\|\widetilde{\mathbf{e}}\|)\rho_\omega(\mathscr{L}_{\widetilde{\mathbf{e}}}) = \rho_\omega\Big(\frac{\gcd(\widetilde{\mathbf{e}})}{\|\widetilde{\mathbf{e}}\|}\mathbb{Z}\Big)\cdot\rho_\omega(\mathscr{L}_{\widetilde{\mathbf{e}}})
\end{aligned}$$

for some shifts $\mathbf{u}_{\widetilde{e}'}\in\operatorname{Span}\mathscr{L}_{\widetilde{\mathbf{e}}}$. It follows that

$$\rho_\omega(\mathscr{L}_{\widetilde{\mathbf{e}}}) \geq \rho_\omega(\mathbb{Z}^m)/\rho_\omega\big(\frac{\gcd(\widetilde{\mathbf{e}})}{\|\widetilde{\mathbf{e}}\|}\mathbb{Z}\big) = (\Theta(\omega))^m/\Theta\Big(\frac{\omega\|\widetilde{\mathbf{e}}\|}{\gcd(\widetilde{\mathbf{e}})}\Big) = 2^{\pm\Theta(m)}\omega^m$$

This implies

$$\max_{\widetilde{\mathbf{x}}\in\mathscr{L}_{\widetilde{\mathbf{e}}}}\tfrac{\rho_\omega(\widetilde{\mathbf{x}}-\mathbf{u}+q\mathscr{L}_{\widetilde{\mathbf{e}}})}{\rho_\omega(-\mathbf{u}+\mathscr{L}_{\widetilde{\mathbf{e}}})} \leq 2^{\Theta(m)}\omega^{-m},$$

hence, $H_\infty(X^q_{\widetilde{\mathbf{e}},\widetilde{e}'}) \geq m\log\omega-\Theta(m)$. Under our assumptions on the parameters, this finally implies $H_\infty(X^q_{\widetilde{\mathbf{e}},\widetilde{e}'})-c_q n\log n=\Omega(n\log n)$, finishing the proof.

Remark 12. Amplifying from $m=\Omega(n\log n)$ samples essentially increases c_s by $1/2$, using $\mathbf{x}$ Gaussian with parameter $\omega=\Theta(1)$. We remark that, using $m=\Omega(n^y)$ with $y>1$ would only lower the constant ω and still incur a loss of $1/2$ in c_s (after some changes in the proof). To reduce this loss from $1/2$ to $1/2-\gamma/2$, we conjecture that we need $m=2^{n^{\mathscr{O}(\gamma)}}$ samples. Further, we need to change the actual amplification procedure to select a sparse (say $0,\pm1$-valued) $\mathbf{x}$ with $\|\mathbf{x}\|_1$ or $\|\mathbf{x}\|_2$ fixed to an appropriate value such as $\|\mathbf{x}\|_1=n^{1-\gamma}$. These changes ensure that $\mathbf{x}$ has just enough (unconditional) min-entropy for $\mathbf{a}'$ to have the correct distribution, but render our proof method inapplicable. If we chose $\mathbf{x}$ as a discrete Gaussian, the probability for $\mathbf{x}=0$ would be too large.

References

1. Yearly report on algorithms and keysizes. *D.SPA.20 Rev. 1.0, ICT-2007-216676 ECRYPT II*, 2012.
2. D. Aggarwal, D. Dadush, O. Regev, and N. Stephens-Davidowitz. Solving the shortest vector problem in 2^n time via discrete gaussian sampling. *CoRR*, 2014.
3. M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of STOC*, pages 601–610, 2001.
4. M. Albrecht, C. Cid, J.-C. Faugére, R. Fitzpatrick, and L. Perret. On the complexity of the bkw algorithm on lwe. *Designs, Codes and Cryptography*, pages 1–30, 2013.
5. M. Albrecht, J.-C. Faugére, R. Fitzpatrick, and L. Perret. Lazy modulus switching for the bkw algorithm on lwe. 8383:429–445, 2014.
6. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Cryptology ePrint Archive, Report 2015/046*, 2015.
7. D. Alexandre, T. Florian, and V. Serge. Better algorithms for lwe and lwr. In *EUROCRYPT*, 2015.
8. Y. Aono, X. Boyen, L. Phong, and L. Wang. Key-private proxy re-encryption under lwe. In *INDOCRYPT 2013*, pages 1–18, 2013.
9. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. 2009.
10. S. Arora and R. Ge. New algorithms for learning in presence of errors. In *Proceedings of the 38th International Colloquim Conference on Automata, Languages and Programming*, ICALP'11, pages 403–415, 2011.
11. L. Babai. On Lovász' lattice reduction and the nearest lattice point problem (shortened version). In *STACS*, pages 13–20, 1985.
12. W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.
13. A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, pages 506–519, 2003.
14. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In *Proceedings of STOC*, pages 575–584, 2013.
15. N. Döttling. Low noise LPN: KDM secure public key encryption and sample amplification. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 604–626, 2015.
16. U. Fincke and M. Pohst. A procedure for determining algebraic integers of given norm. In *Proceedings of EUROCAL*, volume 162 of *Lecture Notes in Computer Science*, pages 194–202, 1983.
17. N. Gama, P. Nguyen, and O. Regev. Lattice enumeration using extreme pruning. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278, 2010.
18. N. Gama and P. Q. Nguyen. Predicting lattice reduction. In *EUROCRYPT*, pages 31–51, 2008.
19. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC '08*, pages 197–206, 2008.
20. O. Goldreich and S. Goldwasser. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540–563, June 2000.
21. O. Goldreich, R. Rubinfeld, and M. Sudan. Learning polynomials with queries: The highly noisy case. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, FOCS, pages 294–303, 1995.

22. Q. Guo, T. Johansson, and P. Stankovski. Coded-bkw: Solving lwe using lattice codes. In *Advances in Cryptology - CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 23–42, 2015.
23. G. Hanrot, X. Pujol, and D. Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. 6841:447–464, 2011.
24. G. Hanrot and D. Stehlé. Improved analysis of kannans shortest lattice vector algorithm. In *Advances in Cryptology - CRYPTO 2007*, volume 4622, pages 170–186, 2007.
25. R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of STOC*, pages 193–206, 1983.
26. R. Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12:415–440, 1987.
27. P. Kirchner and P. Fouque. An improved bkw algorithm for lwe with applications to cryptography and lattices. In *Advances in Cryptology - CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 43–62, 2015.
28. T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, et al. Factorization of a 768-bit rsa modulus. In *Advances in Cryptology–CRYPTO 2010*, pages 333–350. Springer, 2010.
29. T. Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In *CRYPTO*, 2015.
30. A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *J. Cryptology*, 14(4):255–293, 2001.
31. É. Levieil and P.-A. Fouque. An improved lpn algorithm. In R. De Prisco and M. Yung, editors, *Security and Cryptography for Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer Berlin Heidelberg, 2006.
32. R. Lindner and C. Peikert. Better key sizes (and attacks) for lwe-based encryption. In *CT-RSA'11*, pages 319–339, 2011.
33. M. Liu and P. Q. Nguyen. Solving bdd by enumeration: An update. In *CT-RSA*, pages 293–309, 2013.
34. V. Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. volume 3624 of *Lecture Notes in Computer Science*, pages 378–389. 2005.
35. V. Lyubashevsky and D. Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 577–594. 2009.
36. D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*. The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 2002.
37. D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. In *SIAM J. on Computing*, pages 372–381, 2004.
38. D. Micciancio and O. Regev. Lattice-based cryptography. In D. J. Bernstein, J. Buchmann, and E. Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer Berlin Heidelberg, 2009.
39. D. Micciancio and P. Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In *Proceedings of STOC '10*, pages 351–358, 2010.
40. C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of STOC*, pages 333–342, 2009.
41. C. Peikert and D. Micciancio. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.
42. R. Rado. A theorem on the geometry of numbers. *Journal of the London Mathematical Society*, s1-21(1):34–47, 1946.

43. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM Press, 2005.
44. C.-P. Schnorr. Lattice reduction by random sampling and birthday methods. In *STACS*, pages 145–156, 2003.
45. C.-P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In *Math. Programming*, pages 181–191, 1993.