# Practical, Predictable Lattice Basis Reduction *

Daniele Micciancio
UCSD
daniele@eng.ucsd.edu

Michael Walter
UCSD
miwalter@eng.ucsd.edu

**Abstract**

Lattice reduction algorithms are notoriously hard to predict, both in terms of running time and output quality, which poses a major problem for cryptanalysis. While easy to analyze algorithms with good worst-case behavior exist, previous experimental evidence suggests that they are outperformed in practice by algorithms whose behavior is still not well understood, despite more than 30 years of intensive research. This has lead to a situation where a rather complex simulation procedure seems to be the most common way to predict the result of their application to an instance. In this work we present new algorithmic ideas towards bridging this gap between theory and practice. We report on an extensive experimental study of several lattice reduction algorithms, both novel and from the literature, that shows that theoretical algorithms are in fact surprisingly practical and competitive. In light of our results we come to the conclusion that in order to predict lattice reduction, simulation is superfluous and can be replaced by a closed formula using weaker assumptions.

One key technique to achieving this goal is a novel algorithm to solve the Shortest Vector Problem (SVP) in the dual without computing the dual basis. Our algorithm enjoys the same practical efficiency as the corresponding primal algorithm and can be easily added to an existing implementation of it.

## 1 Introduction

Lattice basis reduction is a fundamental tool in cryptanalysis and it has been used to successfully attack many cryptosystems, based on both lattices, and other mathematical problems. (See for example [9, 23, 39, 44–47, 61, 62, 66].) The success of lattice techniques in cryptanalysis is due to a large extent to the fact that reduction algorithms perform much better in practice than predicted by their theoretical worst-case analysis. Basis reduction algorithms have been investigated in many papers over the past 30 years [3, 6, 8, 10, 12–16, 18, 20, 21, 26, 28, 32, 36, 40–42, 45, 48, 50, 51, 54–56, 58–60, 63, 65, 67–69], but the gap between theoretical analysis and practical performance is still largely unexplained. This gap hinders our ability to estimate the security of lattice based cryptographic functions, and it has been widely recognized as one of the main obstacles to the use of lattice cryptography in practice. In this work, we make some modest progress towards this challenging goal.

By and large, the current state of the art in lattice basis reduction (in theory and in practice) is represented by two algorithms:

*Research supported in part by the DARPA SafeWare program and NSF grant CNS-1117936. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or NSF.

- the eminently practical Block-Korkine-Zolotarev (BKZ) algorithm of Schnorr and Euchner [54,60], in its modern BKZ 2.0 incarnation [8] incorporating pruning, recursive preprocessing and early termination strategies [14,18],
- the Slide reduction algorithm of Gama and Nguyen [15], an elegant generalization of LLL [27,40] which provably approximates short lattice vectors within factors related to Mordell's inequality.

Both algorithms make use of a Shortest Vector Problem (SVP) oracle for lower dimensional lattices, and are parameterized by a bound $k$ (called the "block size") on the dimension of these lattices. The Slide reduction algorithm has many attractive features: it makes only a polynomial number of calls to the SVP oracle, all SVP calls are to projected sub-lattices in exactly the same dimension $k$, and it achieves the best known worst-case upper bound on the length of its shortest output vector: $\gamma_k^{(n-1)/(2(k-1))} \det(L)^{1/n}$, where $\gamma_k = \Theta(k)$ is the Hermite constant, and $\det(L)$ is the determinant of the lattice. Unfortunately, it has been reported [15, 16] that in experiments the Slide reduction algorithm is outperformed by BKZ, which produces much shorter vectors for comparable block size. In fact, [15] remarks that even BKZ with block size $k = 20$ produces better reduced bases than Slide reduction with block size $k = 50$. As a consequence, the Slide reduction algorithm is never used in practice, and it has not been implemented and experimentally tested beyond the brief claims made in the initial work [15, 16].[1]

On the other hand, while surprisingly practical in experimental evaluations, the BKZ algorithm has its own shortcomings too. In its original form, BKZ is not even known to terminate after a polynomial number of calls to the SVP oracle, and its observed running time has been reported [16] to grow superpolynomially in the lattice dimension, even when the block size is fixed to some relatively small value $k \approx 30$. Even upon termination, the best provable bounds on the output quality of BKZ are worse than Slide reduction by at least a polynomial factor [15].[2] In practice, in order to address running time issues, BKZ is often employed with an "early termination" strategy [8] that tries to determine heuristically when no more progress is expected from running the algorithm. Theoretical bounds on the quality of the output after a polynomial number of iterations have been proved [18], but they are worse than Slide reduction by even larger polynomial factors. Another fact that complicates the analysis (both in theory and in practice) of the output quality of BKZ is the fact that the algorithm makes SVP calls in all dimensions up to the block size. In theory, this results in a formula that depends on all worst-case (Hermite) constants $\gamma_i$ for $i \leq k$. In practice, the output quality and running time is evaluated by a simulator [8] that initially attempts to numerically estimate the performance of the SVP oracle on random lattices in all possible dimensions up to $k$.

**Our Contribution** We introduce new algorithmic techniques that can be used to design improved lattice basis reduction algorithms, analyze their theoretical performance, implement them, and report on their practical behavior through a detailed set of experiments with block size as high as 75, and several data points per dimension for (still preliminary, but already meaningful) statistical estimation.

[1] While we are referencing two separate works, both refer to the same experimental study.

[2] We remark that polynomial approximation factors, while being asymptotically insignificant, can make a substantial difference in practice, as lattice-based cryptography relies on the hardness of approximating lattice problems within factors that are super-linear in the lattice dimension. In fact, much effort has been put on minimizing such factors in the design of cryptographic constructions [1, 2, 31, 33, 35, 49, 52, 53].

One of our main findings is that the Slide reduction algorithm is much more practical than originally thought, and as the dimension increases, it performs almost as well as BKZ, while at the same time, offering a simple closed-formula to evaluate its output quality. This provides a simple and effective method to evaluate the impact of lattice basis reduction attacks on lattice cryptography, without the need to run simulators or other computer programs [8, 68]. Key to our findings, is a new procedure to enumerate shortest lattice vectors in *dual* lattices, without the need to explicitly compute a dual basis. Interestingly, our dual enumeration procedure is almost identical (syntactically) to the standard enumeration procedure to find short vectors in a (primal) lattice, and, as expected, it is just as efficient in practice. Using our new procedure, we are able to conduct experiments using Slide reduction with significantly larger block size than previously reported, and observe that the gap between theoretical (more predicable) algorithms and practical heuristics gets pretty narrow already for moderate block size and dimension.

For small block sizes (say, up to 40), there is still a substantial gap between the output quality of Slide reduction and BKZ in practice. For this setting, we design a new variant of BKZ, based on lattice duality and a new notion of block reduced basis. Our new DBKZ algorithm can be efficiently implemented using our dual enumeration procedure, achieving running times comparable to BKZ, and matching its experimental output quality for small block size almost exactly. (See Figure 5.) At the same time, our algorithm has various advantages over BKZ, that make it a better target for theoretical analysis: it only makes calls to an SVP oracle in fixed dimension $k$, and it is self dual, in the sense that it performs essentially the same operations when run on a basis or its dual. The fact that all SVP calls on projected sublattices are in the same fixed dimension $k$ has several important implications. First, it results in a simpler bound on the length of the shortest output vector, which can be expressed as a function of just $\gamma_k$. More importantly, this allows to get a practical estimate on the output quality simply by replacing $\gamma_k$ with the value predicted by the Gaussian Heuristic $GH(k)$, commonly used in lattice cryptanalysis. We remark that the $GH(k)$ formula has been validated for moderately large values of $k$, where it gives fairly accurate estimates on the shortest vector length in $k$-dimensional sublattices. However, early work on predicting lattice reduction [16] has also shown that for small $k$ (say, up to $k \leq 25$), BKZ sublattices do not follow the Gaussian Heuristic. As a result, while the BKZ 2.0 simulator of [8] makes extensive use of $GH(k)$ for large values of $k$, it also needs to resort to cumbersome experimental estimations for predicting the result of SVP calls in dimension lower than $k$. By making only SVP calls on $k$-dimensional sublattices, our algorithm obviates the need for any such experimental estimates, and allows to predict the output quality (under the same, or weaker heuristic assumptions than the BKZ 2.0 simulator) just using the GH(k) formula. We stress that this is not only true for the length of the shortest vector found by our algorithm, but one can estimate many more properties of the resulting basis. This is important in many cryptanalytic settings, where lattice reduction is used as a preprocessing for other attacks. In particular, using the Gaussian Heuristic we are able to show that a large part of the basis output by our algorithm can be expected to follow the Geometric Series Assumption [57], an assumption often made about the output of lattice reduction, but so far never proven. (See Section 5 for details.) One last potential advantage of only making SVP calls in fixed dimension $k$ (and, consequently, the ability to use the Gaussian Heuristic for all of them) is that it opens up the possibility of even more accurate stochastic simulations (or analytic solutions) where the GH(k) deterministic formula is replaced by a probability distribution (following the length of the shortest vector in a random $k$-dimensional lattice). We leave the investigation of such a stochastic simulator to future work.

**Technical ideas** Enumeration algorithms (as typically used within block basis reduction) find short vectors in a lattice by examining all possible coordinates $x_1, \ldots, x_n$ of candidate short lattice vectors $\sum_i \mathbf{b}_i \cdot x_i$ with respect to the given lattice basis, and using the length of the projected lattice vector to prune the search. Our dual lattice enumeration algorithm works similarly, but without explicitly computing a basis for the dual lattice. The key technical idea is that one can enumerate over the scalar products $y_i = \langle \mathbf{b}_i, \mathbf{v} \rangle$ of the candidate short dual vectors $\mathbf{v}$ and the primal basis vectors $\mathbf{b}_i$.[3] Perhaps surprisingly, one can also compute the length of the projections of the dual lattice vector $\mathbf{v}$ (required to prune the enumeration tree), without explicitly computing $\mathbf{v}$ or a dual basis. The simplicity of the algorithm is best illustrated just by looking at the pseudo code, and comparing it side-to-side to the pseudo code of standard (primal) lattice enumeration. (See Algorithms 2 and 3 in Section 7.) The two programs are almost identical, leading to a dual enumeration procedure that is just as efficient as primal enumeration, and allowing the application of all standard optimizations (e.g., all various forms of pruning) that have been developed for enumerating in primal lattices.

On the basis reduction front, our DBKZ algorithm is based on a new notion of block-reduced basis. Just as for BKZ, DBKZ-reduction is best described as a recursive definition. In fact, the recursive condition is essentially the same for both algorithms: given a basis $\mathbf{B}$, if $\mathbf{b}$ is a shortest vector in the sublattice generated by the first $k$ basis vectors $\mathbf{B}_{[1,k]}$, we require the projection of $\mathbf{B}$ orthogonal to $\mathbf{b}$ to satisfy the recursive reduction property. The difference between BKZ and DBKZ is that, while BKZ requires $\mathbf{B}_{[1,k]}$ to start with a shortest lattice vector $\mathbf{b} = \mathbf{b}_1$, in DBKZ we require it to *end* with a shortest *dual vector*.[4] This simple twist in the definition of reduced basis leads to a much simpler bound on the length of $\mathbf{b}$, improving the best known bound for BKZ reduction, and matching the theoretical quality of Slide reduction.

**Experiments** To the best of our knowledge, we provide the first experimental study of lattice reduction with large block size parameter beyond BKZ. Even for BKZ we improve on the currently only study involving large block sizes [8] by collecting multiple data points per block size parameter. This allows us to apply standard statistical methods to try to get a sense of the main statistical parameters of the output distribution. Clearly, learning more about the output distribution of these algorithms is highly desirable for cryptanalysis, as an adversary is drawing samples from that distribution and will utilize the most convenient sample, rather than a sample close to the average.

Finally, in contrast to previous experimental work [8, 16], we contribute to the community by making our code[5] and data[6] publicly available. To the best of our knowledge, this includes the first publicly available implementation of dual SVP reduction and Slide reduction. At the time of publication of this work, a modified version of our implementation of dual SVP reduction has been integrated into the main branch of `fpLLL` [4]. We hope that this will spur more research into the predictability of lattice reduction algorithms.

[3] By definition of dual lattice, all these products $y_i$ are integers, and, in fact, they are the coordinates of $\mathbf{v}$ with respect to the standard *dual basis* of $\mathbf{b}_1, \ldots, \mathbf{b}_n$.

[4] To be precise, we require $\mathbf{b}_k^*/\|\mathbf{b}_k^*\|^2$ to be a shortest vector in the dual lattice of $\mathbf{B}_{[1,k]}$. See Section 3 for details.

[5] `http://cseweb.ucsd.edu/~miwalter/src/fplll-dual_enum/fplll-dual_enum.zip`

[6] `http://cseweb.ucsd.edu/~miwalter/src/fplll-dual_enum/results.zip`

## 2 Preliminaries

**Notation** Numbers and reals are denoted by lower case letters. For $n \in \mathbb{Z}_+$ we denote the set $\{0, \ldots, n\}$ by $[n]$. For vectors we use bold lower case letters and the $i$-th entry of a vector $\mathbf{v}$ is denoted by $v_i$. Let $\langle \mathbf{v}, \mathbf{w} \rangle = \sum_i v_i \cdot w_i$ be the scalar product of two vectors. If $p \geq 1$ we define the $p$ norm of a vector $\mathbf{v}$ to be $\|\mathbf{v}\|_p = (\sum |v_i|^p)^{1/p}$. We will only be concerned with the norms given by $p = 1$, 2, and $\infty$. Whenever we omit the subscript $p$, we mean the standard Euclidean norm, i.e. $p = 2$. We define the projection of a vector $\mathbf{b}$ orthogonal to a vector $\mathbf{v}$ as $\pi_{\mathbf{v}}(\mathbf{b}) = \mathbf{b} - \frac{\langle \mathbf{b}, \mathbf{v} \rangle}{\|\mathbf{v}\|^2}\mathbf{v}$. Matrices are denoted by bold upper case letters. The $i$-th column of a matrix $\mathbf{B}$ is denoted by $\mathbf{b}_i$. Furthermore, we denote the submatrix comprising the columns from the $i$-th to the $j$-th column (inclusive) as $\mathbf{B}_{[i,j]}$ and the horizontal concatenation of two matrices $\mathbf{B}_1$ and $\mathbf{B}_2$ by $[\mathbf{B}_1|\mathbf{B}_2]$. For any matrix $\mathbf{B}$ and $p \geq 1$ we define the induced norm to be $\|\mathbf{B}\|_p = \max_{\|\mathbf{x}\|_p=1}(\|\mathbf{B}\mathbf{x}\|_p)$. For $p = 1$ (resp. $\infty$) this is often denoted by the column (row) sum norm; for $p = 2$ this is also known as the spectral norm. It is a classical fact that $\|\mathbf{B}\|_2 \leq \sqrt{\|\mathbf{B}\|_1 \|\mathbf{B}\|_\infty}$. Finally, we extend the projection operator to matrices, where $\pi_{\mathbf{V}}(\mathbf{B})$ is the matrix obtained by applying $\pi_{\mathbf{V}}$ to every column $\mathbf{b}_i$ of $\mathbf{B}$ and $\pi_{\mathbf{V}}(\mathbf{b}_i) = \pi_{\mathbf{v}_k}(\cdots(\pi_{\mathbf{v}_1}(\mathbf{b}_i))\cdots)$.

### 2.1 Lattices

A *lattice* $\Lambda$ is a discrete subgroup of $\mathbb{R}^m$ and is generated by a matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$, i.e. $\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{B}\mathbf{x} \colon \mathbf{x} \in \mathbb{Z}^n\}$. If $\mathbf{B}$ has full column rank, it is called a *basis* of $\Lambda$ and $\dim(\Lambda) = n$ is the dimension (or rank) of $\Lambda$. A lattice has infinitely many bases, which are related to each other by right-multiplication with unimodular matrices. With each matrix $\mathbf{B}$ we associate its *Gram-Schmidt-Orthogonalization* (GSO) $\mathbf{B}^*$, where the $i$-th column $\mathbf{b}_i^*$ of $\mathbf{B}^*$ is defined as $\mathbf{b}_i^* = \pi_{\mathbf{B}^*_{[1,i-1]}}(\mathbf{b}_i) = \mathbf{b}_i - \sum_{j<i} \mu_{i,j}\mathbf{b}_j^*$ and $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$ (and $\mathbf{b}_1^* = \mathbf{b}_1$). For every lattice basis there are infinitely many bases that have the same GSO vectors $\mathbf{b}_i^*$, among which there is a (not necessarily unique) basis that minimizes $\|\mathbf{b}_i\|$ for all $i$. Transforming a basis into this form is commonly known as *size reduction* and is easily and efficiently done using a slight modification of the Gram-Schmidt process. In this work we will implicitly assume all bases to be size reduced. The reader can simply assume that any basis operation described in this work is followed by a size reduction. For a fixed matrix $\mathbf{B}$ we extend the projection operation to indices: $\pi_i(\cdot) = \pi_{\mathbf{B}^*_{[1,i-1]}}(\cdot)$, so $\pi_1(\mathbf{B}) = \mathbf{B}$. Whenever we refer to the *shape* of a basis $\mathbf{B}$, we mean the vector $(\|\mathbf{b}_i^*\|)_{i \in [n]}$. We define $\mathbf{D}^\dagger$ to be the GSO of $\mathbf{D}$ in reverse order.

For every lattice $\Lambda$ there are a few invariants associated to it. One of them is its determinant $\det(\mathcal{L}(\mathbf{B})) = \prod_i \|\mathbf{b}_i^*\|$ for any basis $\mathbf{B}$. Even though the basis of a lattice is not uniquely defined, the determinant is and it is efficiently computable given a basis. Furthermore, for every lattice $\Lambda$ we denote the length of its shortest non-zero vector (also known as the *first minimum*) by $\lambda_1(\Lambda)$, which is always well defined. We use the short-hand notations $\det(\mathbf{B}) = \det(\mathcal{L}(\mathbf{B}))$ and $\lambda_1(\mathbf{B}) = \lambda_1(\mathcal{L}(\mathbf{B}))$. Minkowski's theorem is a classic result that relates the first minimum to the determinant of a lattice. It states that $\lambda_1(\Lambda) \leq \sqrt{\gamma_n}\det(\Lambda)^{1/n}$, for any $\Lambda$ with $\dim(\Lambda) = n$, where $\Omega(n) \leq \gamma_n \leq n$ is Hermite's constant. Finding a (even approximate) shortest nonzero vector in a lattice, commonly known as the *Shortest Vector Problem* (SVP), is NP-hard under randomized reductions [25, 34].

For every lattice $\Lambda$, its *dual* is defined as $\hat{\Lambda} = \{\mathbf{w} \in \mathrm{span}(\Lambda) | \langle \mathbf{w}, \mathbf{v} \rangle \in \mathbb{Z} \text{ for all } \mathbf{v} \in \Lambda\}$. It is a classical fact that $\det(\hat{\Lambda}) = \det(\Lambda)^{-1}$. For a lattice basis $\mathbf{B}$, let $\mathbf{D}$ be the unique matrix that

satisfies $\text{span}(\mathbf{B}) = \text{span}(\mathbf{D})$ and $\mathbf{B}^T\mathbf{D} = \mathbf{D}^T\mathbf{B} = \mathbf{I}$. Then $\widehat{\mathcal{L}(\mathbf{B})} = \mathcal{L}(\mathbf{D})$ and we denote $\mathbf{D}$ as the *dual basis* of $\mathbf{B}$. It follows that for any vector $\mathbf{w} = \mathbf{D}\mathbf{x}$ we have that $\mathbf{B}^T\mathbf{w} = \mathbf{x}$, i.e. we can recover the coefficients $\mathbf{x}$ of $\mathbf{w}$ with respect to the dual basis $\mathbf{D}$ by multiplication with the transpose of the primal basis $\mathbf{B}^T$. Given a lattice basis, its dual basis is computable in polynomial time, but requires at least $\Omega(n^3)$ bit operations using matrix inversion. Finally, if $\mathbf{D}$ is the dual basis of $\mathbf{B}$, their GSOs are related by $\|\mathbf{b}_i^*\| = 1/\|\mathbf{d}_i^\dagger\|$.

In this work we will often modify a lattice basis $\mathbf{B}$ such that its first vector satisfies $\alpha\|\mathbf{b}_1\| \leq \lambda_1(\mathbf{B})$ for some $\alpha \leq 1$. We will call this process *SVP reduction* of $\mathbf{B}$. Given an SVP oracle, it can be accomplished by using the oracle to find the shortest vector in $\mathcal{L}(\mathbf{B})$, prepending it to the basis, and running LLL (cf. Section 2.3) on the resulting generating system. Furthermore, we will modify a basis $\mathbf{B}$ such that its dual $\mathbf{D}$ satisfies $\alpha\|\mathbf{d}_n\| \leq \lambda_1(\widehat{\mathcal{L}(\mathbf{B})})$, i.e. its reversed dual basis is SVP reduced. This process is called *dual SVP reduction*. Note that if $\mathbf{B}$ is dual SVP reduced, then $\|\mathbf{b}_n^*\|$ is maximal among all bases of $\mathcal{L}(\mathbf{B})$. The obvious way to achieve dual SVP reduction is to compute the dual of the basis, SVP reduce it as described above, and compute the primal basis. We present an alternative way to achieve this in Section 7. In the context of reduction algorithms, the relaxation factor $\alpha$ is usually needed for proofs of termination or running time and only impacts the analysis of the output quality in lower order terms. In this work, we will sweep it under the rug and take it implicitly to be a constant close to 1. Finally, we will apply SVP and dual SVP reduction to projected blocks of a basis $\mathbf{B}$, for example we will (dual) SVP reduce the block $\pi_i(\mathbf{B}_{[i,i+k]})$. By that we mean that we will modify $\mathbf{B}$ in such a way that $\pi_i(\mathbf{B}_{[i,i+k]})$ is (dual) SVP reduced. This can easily be achieved by applying the transformations to the original basis vectors instead of their projections.

### 2.2 Enumeration Algorithms

In order to solve SVP in practice, enumeration algorithms are usually employed, since these are the most efficient algorithms for currently realistic dimensions. The standard enumeration procedure, usually attributed to Fincke, Pohst [11], and Kannan [24] can be described as a recursive algorithm: given as input a basis $\mathbf{B} \in \mathbb{Z}^{m\times n}$ and a radius $r$, it first recursively finds all vectors $\mathbf{v}' \in \mathcal{L}(\pi_2(\mathbf{B}))$ with $\|\mathbf{v}'\| \leq r$, and then for each of them finds all $\mathbf{v} \in \mathcal{L}(\mathbf{B})$, s.t. $\pi_2(\mathbf{v}) = \mathbf{v}'$ and $\|\mathbf{v}\| \leq r$, using $\mathbf{b}_1$. This essentially corresponds to a breadth first search on a large tree, where layers correspond to basis vectors and the nodes to the respective coefficients. While it is conceptually simpler to think of enumeration as a BFS, implementations usually employ a depth first search for performance reasons. Pseudo code can be found in Algorithm 3 in Section 7.

There are several practical improvements of this algorithm collectively known as *SchnorrEuchner enumeration* [60]: First, due to the symmetry of lattices, we can reduce the search space by ensuring that the last non zero coefficient is always positive. Furthermore, if we find a vector shorter than the bound $r$, we can update the latter. And finally, we can enumerate the coefficients of a basis vector in order of the length of the resulting (projected) vector and thus increase the chance of finding some short vector early, which will update the bound $r$ and keep the search space smaller.

It has also been demonstrated [14] that reducing the search space (and thus the success probability) – a technique known as pruning – can speed up enumeration by exponential factors. For more details on recent improvements we refer to [14, 19, 20, 36, 69].

### 2.3 Lattice Reduction

As opposed to exact SVP algorithms, lattice reductions approximate the shortest vector. The quality of their output is usually measured in the length of the shortest vector they are able to find with respect to the root determinant of the lattice. This quantity is denoted by the *Hermite factor* $\bar{\delta} = \|\mathbf{b}_1\| / \det(\mathbf{B})^{1/n}$. The Hermite factor depends on the lattice dimension $n$, but the experiments of [16] suggest that the *root Hermite factor* $\delta = \bar{\delta}^{1/n}$ converges to a constant as $n$ increases for popular reduction algorithms. During our experiments we found that to be true at least for large enough dimensions ($n \geq 140$).

The *LLL* algorithm [27] is a polynomial time basis reduction algorithm. A basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$ can be defined to be LLL reduced if $\mathbf{B}_{[1,2]}$ is SVP reduced and $\pi_2(\mathbf{B})$ is LLL reduced. From this it is straight forward to prove that LLL reduction achieves a root Hermite factor of at most $\delta \leq \gamma_2^{1/4} \approx 1.0746$. However, LLL has been reported to behave much better in practice [16, 43].

*BKZ* [54] is a generalization of LLL to larger block size. A basis $\mathbf{B}$ is BKZ reduced with block size $k$ (denoted by BKZ-$k$) if $\mathbf{B}_{[1,\min(k,n)]}$ is SVP reduced and $\pi_2(\mathbf{B})$ is BKZ-$k$ reduced. BKZ achieves this by simply scanning the basis from left to right and SVP reducing each projected block of size $k$ (or smaller once it reaches the end) by utilizing a SVP oracle for all dimensions $\leq k$. It iterates this process (which is usually called a *tour*) until no more change occurs. When $k = n$, this is usually referred to as *HKZ* reduction and is essentially equivalent to solving SVP. The following bound for the Hermite factor holds for $\mathbf{b}_1$ of a BKZ-$k$ reduced basis [18]:

$$\|\mathbf{b}_1\| \leq 2\gamma_k^{\frac{n-1}{2(k-1)}+\frac{3}{2}} \det(\mathbf{B})^{1/n} \tag{1}$$

Equation (1) shows that the root Hermite factor achieved by BKZ-$k$ is at most $\lesssim \gamma_k^{\frac{1}{2(k-1)}}$. Furthermore, while there is no polynomial bound on the number of calls BKZ makes to the SVP oracle, Hanrot, Pujol, and Stehlé showed in [18] that one can terminate BKZ after a polynomial number of calls to the SVP oracle and still provably achieve the bound (1). Finally, BKZ has been repeatedly reported to behave very well in practice [8, 16]. For these reasons, BKZ is very popular in practice and implementations are readily available in different libraries, e.g. in `NTL` [64] or `fpLLL` [4].

In [15], Gama and Nguyen introduced a different block reduction algorithm, namely *Slide reduction*. It is also parameterized by a block size $k$, which is required to divide the lattice dimension $n$, but uses a SVP oracle only in dimension $k$. [7] A basis $\mathbf{B}$ is defined to be slide reduced, if $\mathbf{B}_{[1,k]}$ is SVP reduced, $\pi_2(\mathbf{B}_{[2,k+1]})$ is dual SVP reduced (if $k > n$), and $\pi_{k+1}(\mathbf{B}_{[k+1,n]})$ is slide reduced. Slide reduction, as described in [15], reduces a basis by first alternately SVP reducing all blocks $\pi_{ik+1}(\mathbf{B}_{[ik+1,(i+1)k]})$ and running LLL on $\mathbf{B}$. Once no more changes occur, the blocks $\pi_{ik+2}(\mathbf{B}_{[ik+2,(i+1)k+1]})$ are dual SVP reduced. This entire process is iterated until no more changes occur. Upon termination, the basis is guaranteed to satisfy

$$\|\mathbf{b}_1\| \leq \gamma_k^{\frac{n-1}{2(k-1)}} \det(\mathbf{B})^{1/n} \tag{2}$$

This is slightly better than Equation (1), but the achieved root Hermite factor is also only guaranteed to be less than $\gamma_k^{\frac{1}{2(k-1)}}$. Slide reduction has the desirable properties of only making a

[7] Strictly speaking, the algorithm as described in [15] uses HKZ reduction and thus requires an SVP oracle in lower dimensions as well. However, the entire analysis in [15] only relies on the SVP reducedness of the projected blocks and thus the HKZ reduction can be replaced by SVP reduction, which we do in the following.

polynomial number of calls to the SVP oracle and that all calls are in dimension $k$ (and not in lower dimensions). The latter allows for a cleaner analysis, for example when combined with the Gaussian Heuristic (cf. Section 2.4). Unfortunately, Slide reduction has been reported to be greatly inferior to BKZ in experiments [16], so it is rarely used in practice and we are not aware of any publicly available implementation.

### 2.4 The Gaussian Heuristic

The Gaussian Heuristic gives an approximation of the number of lattice points in a "nice" subset of $\mathbb{R}^n$. More specifically, it says that for a given set $S$ and a lattice $\Lambda$, we have $|S \cap \Lambda| \approx \mathrm{vol}(S)/\det(\Lambda)$. The heuristic has been proved to be very useful in the average case analysis of lattice algorithms. For example, it can be used to estimate the complexity of enumeration algorithms [14, 19] or the output quality of lattice reduction algorithms [8]. For the latter, note that reduction algorithms work by repeatedly computing the shortest vector in some lattice and inserting this vector in a certain position of the basis. To estimate the effect such a step has on the basis, it is useful to be able to predict how long such a vector might be. This is where the Gaussian Heuristic comes in: using the above formula, one can estimate how large the radius of an $n$-dimensional ball (this is the "nice" set) needs to be such that we can expect it to contain a non-zero lattice point (where $n = \dim(\Lambda)$). Using the volume formula for the $n$-dimensional ball, we get an estimate for the shortest non-zero vector in a lattice $\Lambda$:

$$GH(\Lambda) = \frac{(\Gamma(n/2+1) \cdot \det(\Lambda))^{1/n}}{\sqrt{\pi}} \tag{3}$$

If $k$ is an integer, we define $GH(k)$ to be the Gaussian Heuristic (i.e. Equation (3)) for $k$-dimensional lattices with unit determinant. The heuristic has been tested experimentally [14], also in the context of lattice reduction [8, 16], and been found to be too rough in small dimensions, but to be quite accurate starting in dimension $> 45$. In fact, for a precise definition of random lattices (which we are not concerned with in this work) it can be shown that the expected value of the first minimum of the lattice (over the choice of the lattice) converges to Equation (3) as the lattice dimension tends to infinity.[8]

**Heuristic 1** [Gaussian Heuristic] *For a given lattice $\Lambda$, $\lambda_1(\Lambda) = GH(\Lambda)$.*

Invoking Heuristic 1 for all projected sublattices that the SVP oracle is called on during the process, the root Hermite factor achieved by lattice reduction (usually with regards to BKZ) is commonly estimated to be [5]

$$\delta \approx GH(k)^{\frac{1}{k-1}}. \tag{4}$$

However, since the Gaussian Heuristic only seems to hold in large enough dimensions and BKZ makes calls to SVP oracles in all dimensions up to the block size $k$, it is not immediately clear how justified this estimation is. While there is a proof by Chen [7] that under the Gaussian Heuristic, Equation (4) is accurate for BKZ, this is only true as the lattice dimension tends to infinity. It might be reasonable to assume that this also holds in practice as long as the lattice dimension is

[8] One can also formulate Heuristic 1 for a given lattice by assuming it "behaves like a random lattice". Depending on the exact definition of what it means for a lattice to "behave like a random lattice", this version is either stronger as or equivalent to Heuristic 1.

large enough compared to the block size, but in practice and cryptanalytic settings this is often not the case. In fact, in order to achieve an approximation good enough to break a cryptosystem, a block size at least linear in the lattice dimension is often required. As another approach to predicting the output of BKZ, Chen and Nguyen proposed a simulation routine [8]. Unfortunately, the simulator approach has several drawbacks. Obviously, it requires more effort to apply than a closed formula like (4), since it needs to be implemented and "typical" inputs need to be generated or synthesized (among others, the shape of a "typical" HKZ reduced basis in dimension 45). On top of that, the accuracy of the simulator is based on several additional heuristic assumptions, the validity of which has not been independently verified.

To the best of our knowledge there have been no attempts to make similar predictions for Slide reduction, as it is believed to be inferior to BKZ and thus usually not considered for cryptanalysis.

# 3 Self-Dual BKZ

In this section we describe our new reduction algorithm. Like BKZ it is parameterized by a block size $k$ and a SVP oracle in dimension $k$, and acts on the input basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$ by iterating tours. The beginning of every tour is exactly like a BKZ tour, i.e. SVP reducing every block $\pi_i(\mathbf{B}_{[i,i+k-1]})$ from $i = 1$ to $n - k + 1$. We will call this part a *forward tour*. For the last block, which BKZ simply HKZ reduces and where most of the problems for meaningful predictions stem from, we do something different. Instead, we dual SVP the last block and proceed by dual SVP reducing all blocks of size $k$ backwards (which is a *backward tour*). After iterating this process (which we call a *tour* of Self-Dual BKZ) the algorithm terminates when no more progress is made. The algorithm is formally described in Algorithm 1.

**Algorithm 1** Self-Dual BKZ

`procedure` DBKZ ($\mathbf{B}$, $k$, $\mathsf{SVP}_k$)
**Input:** A lattice basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$, a block size $k$, a SVP oracle in dimension $k$
**Output:** A $k$-reduced basis $\mathbf{B}'$ (See Definition 1 for a formal definition.)

```
1   do
2     for i = 1 ... n − k
3       SVP reduce π_i(B_[i,i+k−1]) using SVP_k
4     for i = n − k + 1 ... 1
5       dual SVP reduce π_i(B_[i,i+k−1]) using SVP_k
6   while progress is made
7   return B
```

Note that, like BKZ, Self-Dual BKZ (DBKZ) is a proper block generalization of the LLL algorithm, which corresponds to the case $k = 2$.

The terminating condition in Line `6` is left ambiguous at this point on purpose as there are several sensible ways to approach this as we will see in the next section. One has to be careful to, on the one hand guarantee termination, while on the other hand achieving a meaningful reducedness definition.

## 3.1 Analysis

The output of Algorithm 1 satisfies the following reducedness definition upon termination:

**Definition 1** *A basis* $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ *is* $k$*-reduced if either* $n < k$*, or it satisfies the following conditions:*

- $\|\mathbf{b}_k^*\|^{-1} = \lambda_1(\widehat{\mathcal{L}(\mathbf{B}_{[1,k]})})$*, and*
- *for some SVP reduced basis* $\tilde{\mathbf{B}}$ *of* $\mathcal{L}(\mathbf{B}_{[1,k]})$*,* $\pi_2([\tilde{\mathbf{B}}|\mathbf{B}_{[k+1,n]}])$ *is* $k$*-reduced.*

We first prove that Algorithm 1 indeed achieves Definition 1 when used with a specific terminating condition:

**Lemma 1** *Let* $\mathbf{B}$ *be an* $n$*-dimensional basis. If* $\pi_{k+1}(\mathbf{B})$ *is the same before and after one loop of Algorithm 1, then* $\mathbf{B}$ *is* $k$*-reduced.*

*Proof* The proof is inductive: for $n = k$ the result is trivially true. So, assume $n > k$, and that the result already holds for $n - 1$. At the end of each iteration, the first block $\mathbf{B}_{[1,k]}$ is dual-SVP reduced by construction. So, we only need to verify that for some $\tilde{\mathbf{B}}$ an SVP reduced basis for $\mathcal{L}(\mathbf{B}_{[1,k)})$, the projection $\pi_2([\tilde{\mathbf{B}}|\mathbf{B}_{[k+1,n]}])$ is also $k$-reduced. Let $\tilde{\mathbf{B}}$ be the SVP reduced basis produced in the first step. Note that the first and last operation in the loop do not change $\mathcal{L}(\mathbf{B}_{[1,k]})$ and $\mathbf{B}_{[k+1,n]}$. It follows that $\pi_{k+1}(\mathbf{B})$ is the same before and after the partial tour (the tour without the first and the last step) on the projected basis $\pi_2([\tilde{\mathbf{B}}|\mathbf{B}_{[k+1,n]}])$, and so $\pi_{k+2}(\mathbf{B})$ is the same before and after the partial tour. By induction hypothesis, $\pi_2([\tilde{\mathbf{B}}|\mathbf{B}_{[k+1,n]}])$ is $k$-reduced. □

Lemma 1 gives a terminating condition which ensures that the basis is reduced. We remark that it is even possible to adapt the proof such that it is sufficient to check that the shape of the projected basis $\pi_{k+1}(\mathbf{B})$ is the same before and after the tour, which is much closer to what one would do in practice to check if *progress was made* (cf. Line 6). However, this requires to relax the definition of SVP-reduction slightly, such that the first vector is not necessarily a shortest vector, but merely a short vector achieving Minkowski's bound. Since this is the only property of SVP reduced bases we need for the analysis below, this does not affect the worst case output quality. Finally, we are aware that it is not obvious that either of these conditions are ever met, e.g. (the shape of) $\pi_{k+1}(\mathbf{B})$ might loop indefinitely. However, in Section 4 we show that one can put a polynomial upper bound on the number of loops without sacrificing worst case output quality.

To show that the output quality of Self-Dual BKZ in the worst case is at least as good as BKZ's worst case behavior, we analyze the Hermite factor it achieves:

**Theorem 1** *If* $\mathbf{B}$ *is* $k$*-reduced, then* $\lambda_1(\mathbf{B}_{[1,k]}) \leq \sqrt{\gamma_k}^{\frac{n-1}{k-1}} \cdot \det(\mathbf{B})^{1/n}$.

*Proof* Assume without loss of generality that $\mathcal{L}(\mathbf{B})$ has determinant 1, and let $\Delta$ be the determinant of $\mathcal{L}(\mathbf{B}_{[1,k]})$. Let $\lambda \leq \sqrt{\gamma_k}\Delta^{1/k}$ and $\hat{\lambda} \leq \sqrt{\gamma_k}\Delta^{-1/k}$ be the lengths of the shortest nonzero primal and dual vectors of $\mathcal{L}(\mathbf{B}_{[1,k]})$. We need to prove that $\lambda \leq \sqrt{\gamma_k}^{\frac{n-1}{k-1}}$.

We first show, by induction on $n$, that the determinant $\Delta_1$ of the first $k-1$ vectors is at most $\sqrt{\gamma_k}^{n-k+1}\det(\mathbf{B})^{(k-1)/n} = \sqrt{\gamma_k}^{n-k+1}$. Since $\mathbf{B}$ is $k$-reduced, this determinant equals $\Delta_1 = \hat{\lambda} \cdot \Delta \leq \sqrt{\gamma_k}\Delta^{1-1/k}$. (This alone already proves the base case of the induction for $n = k$.) Now, let $\tilde{\mathbf{B}}$ be a SVP reduced basis of $\mathcal{L}(\mathbf{B}_{[1,k]})$ satisfying the $k$-reduction definition, and consider the determinant $\Delta_2 = \Delta/\lambda$ of $\pi_2(\tilde{\mathbf{B}})$. Since $\pi_2([\tilde{\mathbf{B}}|\mathbf{B}_{[k+1,n]}])$ has determinant $1/\|\tilde{\mathbf{b}}_1\| = 1/\lambda$, by induction hypothesis we have $\Delta_2 \leq \sqrt{\gamma_k}^{n-k}(1/\lambda)^{(k-1)/(n-1)}$.

$$\Delta = \lambda\Delta_2 \leq \sqrt{\gamma_k}^{n-k}\lambda^{\frac{n-k}{n-1}} \leq \sqrt{\gamma_k}^{n-k}(\sqrt{\gamma_k}\Delta^{\frac{1}{k}})^{\frac{n-k}{n-1}} = \sqrt{\gamma_k}^{\frac{(n-k)n}{n-1}}\Delta^{\frac{n-k}{k(n-1)}}.$$

Rising both sides to the power $(n-1)/n$ we get $\Delta^{1-\frac{1}{n}} \leq \sqrt{\gamma_n}^{n-k}\Delta^{\frac{1}{k}-\frac{1}{n}}$, or, equivalently, $\Delta^{1-\frac{1}{k}} \leq \sqrt{\gamma_k}^{n-k}$. It follows that $\Delta_1 = \hat{\lambda}\Delta \leq \sqrt{\gamma_k}\Delta^{1-\frac{1}{k}} \leq \sqrt{\gamma_k}^{n-k+1}$, concluding the proof by induction.

We can now prove the main theorem statement. Recall from the inductive proof that $\Delta \leq \sqrt{\gamma_k}^{n-k}\lambda^{\frac{n-k}{n-1}}$. Therefore, $\lambda \leq \sqrt{\gamma_k}\Delta^{1/k} \leq \sqrt{\gamma_k}^{\frac{n}{k}}\lambda^{\frac{n-k}{k(n-1)}}$. Solving for $\lambda$, proves the theorem. □

# 4 Dynamical System

Proving a good running time on DBKZ directly seems just as hard as for BKZ, so in this section we analyze the DBKZ algorithm using the dynamical system technique from [18].

Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ be an input basis to DBKZ, and assume without loss of generality that $\det(\mathbf{B}) = 1$. During a forward tour, our algorithm computes a sequence of lattice vectors $\mathbf{B}' = [\mathbf{b}'_1, \ldots, \mathbf{b}'_{n-k}]$ where each $\mathbf{b}'_i$ is set to a shortest vector in the projection of $[\mathbf{b}_i, \ldots, \mathbf{b}_{i+k-1}]$ orthogonal to $[\mathbf{b}'_1, \ldots, \mathbf{b}'_{i-1}]$. This set of vectors can be extended to a basis $\mathbf{B}'' = [\mathbf{b}''_1, \ldots, \mathbf{b}''_n]$ for the original lattice. Since $[\mathbf{b}'_1, \ldots, \mathbf{b}'_{i-1}]$ generates a primitive sublattice of $[\mathbf{b}_i, \ldots, \mathbf{b}_{i+k-1}]$, the projected sublattice has determinant $\det(\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{i+k-1}))/\det(\mathcal{L}(\mathbf{b}'_1, \ldots, \mathbf{b}'_{i-1}))$, and the length of its shortest vector is

$$\|(\mathbf{b}'_i)^*\| \leq \sqrt{\gamma_k}\left(\frac{\det(\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{i+k-1}))}{\det(\mathcal{L}(\mathbf{b}'_1, \ldots, \mathbf{b}'_{i-1}))}\right)^{1/k}. \tag{5}$$

At this point, simulations based on the Gaussian Heuristics typically assume that (5) holds with equality. In order to get a rigorous analysis without heuristic assumptions, we employ the amortization technique of [18, 19]. For every $i = 1, \ldots, n-k$, let $x_i = \log\det(\mathbf{b}_1, \ldots, \mathbf{b}_{k+i-1})$ and $x'_i = \log\det(\mathbf{b}'_1, \ldots, \mathbf{b}'_i)$. Using (5), we get for all $i = 1, \ldots, n-k$,

$$\begin{aligned} x'_i &= x'_{i-1} + \log\|(\mathbf{b}'_i)^*\| \\ &\leq x'_{i-1} + \alpha + \frac{x_i - x'_{i-1}}{k} \\ &= \omega x'_{i-1} + \alpha + (1-\omega)x_i \end{aligned}$$

where $\omega = (1 - 1/k)$, $\alpha = \frac{1}{2}\log\gamma_k$ and $x'_0 = 0$. By induction on $i$,

$$x'_i \leq \alpha\frac{1-\omega^i}{1-\omega} + (1-\omega)\sum_{j=1}^{i}\omega^{i-j}x_j,$$

or, in matrix notation $\mathbf{x}' \leq \mathbf{b} + \mathbf{A}\mathbf{x}$ where

$$\mathbf{b} = \alpha k\begin{bmatrix} 1-\omega \\ \vdots \\ 1-\omega^{n-k} \end{bmatrix} \qquad \mathbf{A} = \frac{1}{k}\begin{bmatrix} 1 & & & \\ \omega & 1 & & \\ \vdots & \ddots & \ddots & \\ \omega^{n-k-1} & \cdots & \omega & 1 \end{bmatrix}.$$

Since all the entries of $\mathbf{A}$ are positive, we also see that if $X_i \geq x_i$ are upper bounds on the initial values $x_i$ for all $i$, then the vector $X' = \mathbf{A}X + \mathbf{b}$ gives upper bounds on the output values $x'_i \leq X'_i$.

The vector $\mathbf{x}'$ describes the shape of the basis matrix before the execution of a backward tour. Using lattice duality, the backward tour can be equivalently formulated by the following steps:

1. Compute the reversed dual basis $\mathbf{D}$ of $\mathbf{B}'$
2. Apply a forward tour to $\mathbf{D}$ to obtain a new dual basis $\mathbf{D}'$
3. Compute the reversed dual basis of $\mathbf{D}'$

The reversed dual basis computation yields a basis $\mathbf{D}$ such that, for all $i = 1, \ldots, n-k$,

$$\begin{aligned} y_i &= \log\det(\mathbf{d}_1, \ldots, \mathbf{d}_{k+i-1}) \\ &= -\log(\det(\mathbf{B}')/\det([\mathbf{b}'_1, \ldots, \mathbf{b}'_{n-k+1-i}])) \\ &= \log\det([\mathbf{b}'_1, \ldots, \mathbf{b}'_{n-k+1-i}]) = x'_{n-k+1-i}. \end{aligned}$$

So, the vector $\mathbf{y}$ describing the shape of the dual basis at the beginning of the backward tour is just the reverse of $\mathbf{x}'$. It follows that applying a full (forward and backward) DBKZ tour produces a basis such that if $X$ are upper bounds on the log determinants $\mathbf{x}$ of the input matrix, then the log determinants of the output matrix are bounded from above by

$$\mathbf{R}(\mathbf{A}\mathbf{R}(\mathbf{A}X + \mathbf{b}) + \mathbf{b}) = (\mathbf{R}\mathbf{A})^2 X + (\mathbf{R}\mathbf{A} + \mathbf{I})\mathbf{R}\mathbf{b}$$

where $\mathbf{R}$ is the coordinate reversal permutation matrix. This leads to the study of the discrete time affine dynamical system

$$X \mapsto (\mathbf{R}\mathbf{A})^2 X + (\mathbf{R}\mathbf{A} + \mathbf{I})\mathbf{R}\mathbf{b}. \tag{6}$$

### 4.1 Output Quality

We first prove that this system has at most one fixed point.

**Claim 1** *The dynamical system (6) has at most one fixed point.*

*Proof* Any fixed point is a solution to the linear system $((\mathbf{R}\mathbf{A})^2 - \mathbf{I})X + (\mathbf{R}\mathbf{A} + \mathbf{I})\mathbf{R}\mathbf{b} = \mathbf{0}$. To prove uniqueness, we show that the matrix $((\mathbf{R}\mathbf{A})^2 - \mathbf{I})$ is non-singular, i.e., if $(\mathbf{R}\mathbf{A})^2\mathbf{x} = \mathbf{x}$ then $\mathbf{x} = \mathbf{0}$. Notice that the matrix $\mathbf{R}\mathbf{A}$ is symmetric, so we have $(\mathbf{R}\mathbf{A})^2 = (\mathbf{R}\mathbf{A})^T\mathbf{R}\mathbf{A} = \mathbf{A}^T\mathbf{A}$. So proving $((\mathbf{R}\mathbf{A})^2 - \mathbf{I})$ is non-singular is equivalent to showing that 1 is not an eigenvalue of $\mathbf{A}^T\mathbf{A}$. We have $\rho(\mathbf{A}^T\mathbf{A}) = \|\mathbf{A}\|_2^2 \leq \|\mathbf{A}\|_1\|\mathbf{A}\|_\infty$, where $\rho(\cdot)$ denotes the spectral radius of the given matrix (i.e. the largest eigenvalue in absolute value). But we also have

$$\|\mathbf{A}\|_\infty = \|\mathbf{A}\|_1 = \frac{1}{k}\sum_{i=0}^{n-k-1}\omega^i = \frac{1-\omega^{n-k}}{k(1-\omega)} = 1 - \omega^{n-k} < 1 \tag{7}$$

which shows that the absolute value of any eigenvalue of $\mathbf{A}^T\mathbf{A}$ is strictly smaller than 1.□

We need to find a fixed point for (6). We have proved that $(\mathbf{R}\mathbf{A})^2 - \mathbf{I}$ is a non-singular matrix. Since $(\mathbf{R}\mathbf{A})^2 - \mathbf{I} = (\mathbf{R}\mathbf{A} + \mathbf{I})(\mathbf{R}\mathbf{A} - \mathbf{I})$, it follows that $(\mathbf{R}\mathbf{A} \pm \mathbf{I})$ are also non singular. So, we can factor $(\mathbf{R}\mathbf{A} + \mathbf{I})$ out of the fixed point equation $((\mathbf{R}\mathbf{A})^2 - \mathbf{I})\mathbf{x} + (\mathbf{R}\mathbf{A} + \mathbf{I})\mathbf{R}\mathbf{b} = \mathbf{0}$, and obtain $(\mathbf{R}\mathbf{A} - \mathbf{I})\mathbf{x} + \mathbf{R}\mathbf{b} = 0$. This shows that the only fixed point of the full dynamical system (if it exists) must also be a fixed point of a forward tour $\mathbf{x} \mapsto \mathbf{R}(\mathbf{A}\mathbf{x} + \mathbf{b})$.

**Claim 2** *The fixed point of the dynamical system* $\mathbf{x} \mapsto \mathbf{R}(\mathbf{A}\mathbf{x} + \mathbf{b})$ *is given by*

$$x_i = \frac{(n-k-i+1)(k+i-1)}{k-1}\alpha. \tag{8}$$

*Proof* The unique fixed point of the system is given by the solution to the linear system $(\mathbf{R}-\mathbf{A})\mathbf{x} = \mathbf{b}$. We prove that (8) is a solution to the system by induction on the rows. For the first row, the system yields

$$x_{n-k} - x_1/k = \alpha. \tag{9}$$

From (8) we get that $x_{n-k} = \frac{n-1}{k-1}\alpha$ and $x_1 = \frac{k(n-k)}{k-1}\alpha$. Substituting these into (9), the validity is easily verified.

The $r$-th row of the system is given by

$$x_{n-k-r+1} - \frac{1}{k}\left(\sum_{j=1}^{r} \omega^{r-j} x_j\right) = \frac{1-\omega^r}{1-\omega}\alpha \tag{10}$$

which is equivalent to

$$x_{n-k-r+1} + \omega\left(x_{n-k-r+2} - \frac{1}{k}\left(\sum_{j=1}^{r-1} \omega^{r-1-j} x_j\right)\right) - \frac{x_r}{k} - \omega x_{n-k-r+2} = \frac{1-\omega^r}{1-\omega}\alpha. \tag{11}$$

By induction hypothesis, this is equivalent to

$$\omega\left(\frac{1-\omega^{r-1}}{1-\omega}\right)\alpha + x_{n-k-r+1} - \frac{x_r}{k} - \omega x_{n-k-r+2} = \frac{1-\omega^r}{1-\omega}\alpha. \tag{12}$$

Substituting (8) in for $i = n-k-r+1$, $r$, and $n-k-r+2$, we get

$$x_{n-k-r+1} - \frac{x_r}{k} - \omega x_{n-k-r+2} = \frac{kr(n-r) - (n-r-k+1)(r+k-1) - (k-1)(r-1)(n-r+1)}{k(k-1)}\alpha$$

which, after some tedious, but straight forward, calculation can be shown to be equal to $\alpha$ (i.e. the fraction simplifies to 1). This in turn shows that the left hand side of (12) is equivalent to

$$\omega\left(\frac{1-\omega^{r-1}}{1-\omega}\right)\alpha + \alpha$$

which is equal to its right hand side. □

Note that since $x_1$ corresponds to the log determinant of the first block, applying Minkowski's theorem results in the same worst case Hermite factor as proved in Theorem 1.

### 4.2 Convergence

Consider any input vector $\mathbf{v}$ and write it as $\mathbf{v} = \mathbf{x} + \mathbf{e}$, where $\mathbf{x}$ is the fixed point of the dynamical system as in (8). The system sends $\mathbf{v}$ to $\mathbf{v} \mapsto \mathbf{R}\mathbf{A}\mathbf{v}+\mathbf{b} = \mathbf{R}\mathbf{A}\mathbf{x}+\mathbf{R}\mathbf{A}\mathbf{e}+\mathbf{b} = \mathbf{x}+\mathbf{R}\mathbf{A}\mathbf{e}$, so the difference $\mathbf{e}$ to the fixed point is mapped to $\mathbf{R}\mathbf{A}\mathbf{e}$ in each iteration. In order to analyze the convergence of the algorithm, we consider the induced norm of the matrix $\|\mathbf{R}\mathbf{A}\|_p = \|\mathbf{A}\|_p$, since after $t$ iterations

the difference is $(\mathbf{RA})^t\mathbf{e}$ and so its norm is bounded by $\|(\mathbf{RA})^t\mathbf{e}\|_p \leq \|(\mathbf{RA})^t\|_p\|\mathbf{e}\|_p \leq \|\mathbf{RA}\|_p^t\|\mathbf{e}\|_p$. So if the induced norm of $\mathbf{A}$ is strictly smaller than 1, the corresponding norm of the error vector follows an exponential decay. While the spectral norm of $\mathbf{A}$ seems hard to bound, the 1 and the infinity norm are straight forward to analyze. In particular, we saw in (7) that $\|\mathbf{A}\|_\infty = 1-\omega^{n-k}$. This proves that the algorithm converges. Furthermore, let the input be a basis $\mathbf{B}$ (with $\det(\mathbf{B}) = 1$), the corresponding vector $\mathbf{v} = (\log\det(\mathbf{b}_1, \ldots, \mathbf{b}_{k+i-1}))_{1\leq i\leq n}$ and write $\mathbf{v} = \mathbf{x} + \mathbf{e}$. Then we have $\|\mathbf{e}\|_\infty = \|\mathbf{v} - \mathbf{x}\|_\infty \leq \|\mathbf{v}\|_\infty + \|\mathbf{x}\|_\infty \leq \mathrm{poly}(n, \mathrm{size}(\mathbf{B}))$. This implies that for

$$t = \mathrm{polylog}(n, \mathrm{size}(\mathbf{B}))/\omega^{n-k} \approx O(e^{(n-k)/k})\mathrm{polylog}(n, \mathrm{size}(\mathbf{B})) \tag{13}$$

we have that $\|(\mathbf{RA})^t\mathbf{e}\| \leq c$ for constant $c$. Equation (13) already shows that for $k = \Omega(n)$, the algorithm converges in a number of tours polylogarithmic in the lattice dimension $n$, i.e. makes at most $\tilde{O}(n)$ SVP calls. In the initial version of this work, proving polynomial convergence for arbitrary $k$ was left as an open problem. Recently, Neumaier filled this gap [38]. In what follows we reformulate his proof using our notation for completeness, but we stress that this is originally Neumaier's work.

**Claim 3** *Let* $\mathbf{x}$ *be the fixed point of the dynamical system as defined in Claim 2 and* $r_i = e_i/x_i$ *be the relative error of the dynamical system. Then, if* $\mathbf{r}, \mathbf{r}'$ *are the relative errors before and after the execution of one iteration of the system, then* $\|\mathbf{r}'\|_\infty \leq (1-\epsilon)\|\mathbf{r}\|_\infty$ *for* $\epsilon = 1/(1+n^2/(4k(k-1))) \approx (2k/n)^2$. *When* $k \geq n/2$, *it is enough to take* $\epsilon = (k-1)/(n-1)$.

*Proof* Assume without loss of generality that $\|\mathbf{r}\|_\infty = (k-1)/\alpha$, i.e., $|e_i| \leq ((k-1)/\alpha)x_i = (n-k-i+1)(k+i-1)$ for all $i = 1, \ldots, n-k$. We need to prove that $|r'_j| = |(\mathbf{RAe})_j|/x_j \leq ((k-1)/\alpha)(1-\epsilon)$ for all $j$, or, equivalently,

$$|(\mathbf{Ae})_j| = |(\mathbf{RAe})_{n-k-j+1}| \leq \frac{k-1}{\alpha}(1-\epsilon)x_{n-k-j+1} = j(n-j)(1-\epsilon).$$

By the definition of $\mathbf{A}$, we have

$$|(\mathbf{Ae})_j| \leq \frac{1}{k}\sum_{i=1}^{j}\omega^{j-i}|e_i| \leq \frac{1}{k}\sum_{i=1}^{j}\omega^{j-i}(n-k-i+1)(k+i-1) \equiv f(j).$$

So, it is enough to prove that the function $f(j)$ on the right hand side satisfies $f(j) \leq j(n-j)(1-\epsilon)$. We prove this inequality by induction on $j$, under the assumption that

$$\epsilon \leq g(j) \equiv \frac{k(k-1)}{k(n-2j-1)+j(n-j)}.$$

- base case ($j = 1$): $f(1) = n-k \leq (n-1)(1-\epsilon)$ if and only if $\epsilon \leq g(0) = (k-1)/(n-1)$.
- inductive step: assume $f(j) \leq j(n-j)(1-\epsilon)$ by inductive hypothesis. We need to prove that

  $$f(j+1) = \omega f(j) + \frac{(n-k-j)(k+j)}{k} \leq \frac{k-1}{k}j(n-j)(1-\epsilon) + \frac{(n-k-j)(k+j)}{k}$$

  is at most $(j+1)(n-(j+1))(1-\epsilon)$. This is true if an only if $\epsilon \leq g(j)$.

This concludes the inductive proof, as long as $\epsilon \leq g(j)$ for $j = 0, \ldots, (n-k)$. Finally, we observe that the function $g(j)$ is minimized at $j = \frac{n}{2} - k$, with minimum $g(\frac{n}{2} - k) = 1/(1 + n^2/(4k(k-1)))$. When $k > n/2$, the minimum is achieved at $j < 0$ and $g(j)$ is monotonically increasing for $j \leq 0$. So it is enough to take $\epsilon = g(0) = (k-1)/(n-1)$.□

By a similar argument as above, this shows that the error can be made arbitrarily close to 0 in $O((n/2k)^2)\text{polylog}(n, \text{size}(\mathbf{B}))$ tours.

# 5 Heuristic Analysis

In the context of cryptanalysis, we are more interested in the average case behavior of algorithms. For this we can use a very simple observation to predict the Hermite factor achieved by DBKZ. Note that the proof of Theorem 1 is based solely on Minkowski's bound $\lambda_1(\mathbf{B}) \leq \sqrt{\gamma_n} \det(\mathbf{B})^{1/n}$. Replacing it with Heuristic 1 yields the following corollary.

**Corollary 1** *Applying Heuristic 1 to every lattice that is passed to the SVP oracle during the execution of Algorithm 1, if* $\mathbf{B}$ *is* $k$*-reduced, then* $\lambda_1(\mathbf{B}_{1,k}) = GH(k)^{\frac{n-1}{k-1}} \det(\mathbf{B})^{1/n}$.

As the Hermite factor is the most relevant quantity in many cryptanalytic settings, Corollary 1 is already sufficient for many intended applications in terms of output quality. We remark that the proof of achieved worst-case output quality of Slide reduction also only relies on Minkowski's bound. This means the same observation can be used to predict the average case behavior of Slide reduction and yields the same estimate as Corollary 1. In fact, from the recursive definition of Slide reduction it is clear that this yields even more information about the returned basis: we can use Corollary 1 to predict the norm of $\|\mathbf{b}_{ik+1}\|$ for all $i \in [n/k]$. A short calculation shows that these vectors follow a geometric series, supporting a frequently assumed behavior of lattice reduction, namely the *Geometric Series Assumption* [57].

However, many attacks [30,45] require to estimate the average case output much more precisely. Fortunately, applying a similar trick as in Corollary 1 to the dynamical systems analysis in Section 4 allows us to obtain much more information about the basis. For this, note that again we can replace Minkowski's theorem in the analysis by Heuristic 1. This transformation changes the dynamical system in (6) only slightly, the only difference being that $\alpha = \frac{1}{2} \log GH(k)$. As the analysis is independent of the constant $\alpha$, we can translate the fixed point in (8) to information about the shape of the basis that DBKZ is likely to return.

**Corollary 2** *Applying Heuristic 1 to every lattice that is passed to the SVP oracle during the execution of Algorithm 1, the fixed point of the heuristic dynamical system, i.e. (6) with* $\alpha = \frac{1}{2} \log GH(k)$*, is (8) with the same* $\alpha$ *and implies that after one more forward tour, the basis satisfies*

$$\|\mathbf{b}_i^*\| = GH(k)^{\frac{n+1-2i}{2(k-1)}} \det(\mathcal{L}(\mathbf{B}))^{\frac{1}{n}} \tag{14}$$

*for all* $i \leq n - k$.

*Proof* According to (8), upon termination of Algorithm 1 the output basis satisfies

$$\log(\det([\mathbf{b}_1, \ldots, \mathbf{b}_i])) = \frac{(n-k-i+1)(k+i-1)}{k-1}\alpha$$

By Heuristic 1 we have $\log \|\mathbf{b}_1\| = \alpha + x_1/k$, from which Equation (14) easily follows for $i = 1$. Now assume (14) holds for all $j < i$. Then we have, again by Heuristic 1, $\log \|\mathbf{b}_i^*\| = \alpha + (x_i - \sum_{j<i} \log \|\mathbf{b}_j^*\|)/k$. Invoking the induction hypothesis, Equation (14) easily follows for all $i \leq n - k$. □

Corollary 2 shows that the output of the DBKZ algorithm, if terminated after a forward tour, can be expected to closely follow the GSA, at least for all $i \leq n - k$ and can be computed using simple closed formulas. It is noteworthy that the self-dual properties of DBKZ imply that if terminated after a backward tour, the GSA holds for all $i \geq k$. This means, depending on the application one can choose which part of the output basis to predict. Moreover, we see that DBKZ allows to predict a much larger part of the basis than Slide reduction solely based on the Gaussian Heuristic. If one is willing to make additional assumptions, i.e. assumptions about the shape of a $k$-dimensional HKZ reduced basis, the BKZ simulator allows to predict the shape of the entire basis output by BKZ. Obviously, the same assumptions can be used to estimate the remaining parts of the shape of the basis in the case of Slide reduction and DBKZ, since a final application of a HKZ reduction to individual blocks of size $k$ only requires negligible amount of time compared to the running time of the entire algorithm. Furthermore, since the estimation of the known part of the shape (from Corollary 2 and 1) do not depend on these additional assumptions, the estimation for Slide reduction and DBKZ is much less sensitive to the (in-)correctness of these assumptions, while errors propagate during the BKZ simulation.

To compare the expected output of BKZ, DBKZ, and Slide reduction, we generated a Goldstein-Mayer lattice [17] in dimension $n = 200$ with numbers of bit size 2000, applied LLL to it, and simulated the execution of BKZ with block size $k = 100$ until no more progress was made. The output in terms of the logarithm of the shape of the basis for the first 100 basis vectors is shown in Figure 1 and compared to the GSA. Recall that the latter represents the expected output of DBKZ and, to some degree, Slide reduction. Under the assumption that Heuristic 1 and the BKZ simulator are accurate, one would expect BKZ to behave a little worse than the other two algorithms in terms of output quality.

# 6 Experiments

For an experimental comparison, we implemented DBKZ and Slide reduction in `fpLLL`. SVP reduction in `fplll` is implemented in the standard way as described in Section 2.1. For dual SVP reduction we used the algorithm explained in the Section 7.

## 6.1 Methodology

In the context of cryptanalysis we are usually interested in the root Hermite factor achievable using lattice reduction in order to choose parameters for cryptosystems, as this often determines the success probability and/or complexity of an attack. It is clear that merely reporting on the average root Hermite factor achieved is of limited use for this. Instead we will view the resulting root Hermite factor achieved by a certain reduction algorithm (with certain parameters) as a random variable and try to estimate the main statistical parameters of its distribution. We believe this will eventually allow for more meaningful security estimates. The only previous experimental work studying properties of the underlying distribution of the root Hermite factor [16] suggests that it is a Gaussian-like but the study is limited to relatively small block sizes.

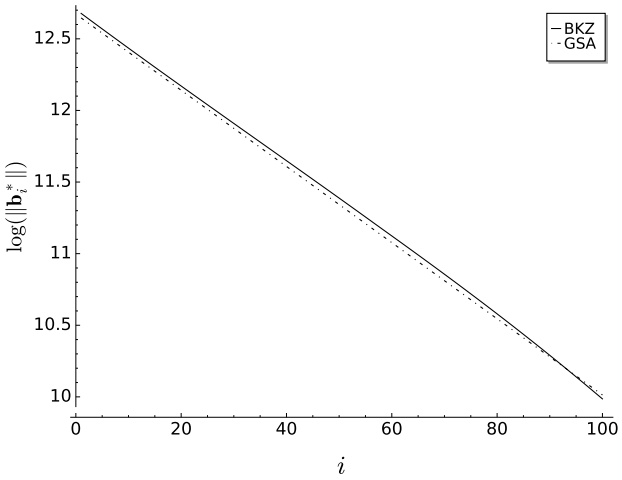



Figure 1: Expected shape of the first 100 basis vectors in dimension $n = 200$ after BKZ compared to the GSA. Note that the latter corresponds exactly to the expected shape of the first 100 basis vectors after DBKZ (cf. 2).

Since experiments with lattice reduction are rather time consuming, it is infeasible to generate as much data as desirable to estimate statistical parameters like the mean value and standard deviation accurately. A standard statistical technique to overcome this is to use bootstrapping to compute confidence intervals for these parameters. Roughly speaking, in order to compute the confidence interval for an estimator from a set of $N$ samples, we sample $l$ sets of size $N$ with replacement from the original samples and compute the estimator for each of them. Intuitively, this should give a sense of the variability of the estimator computed on the samples. Our confidence interval with confidence parameter $\alpha$, according to the bootstrap percentile interval method, is simply the $\alpha/2$ and $1-\alpha/2$ quantiles. For further discussion we refer to [70]. Throughout this work we use $\alpha = .05$ and $l = 100$. The complete confidence intervals for mean value and standard deviation are listed in Appendix B. Whenever we refer to the standard deviation of a distribution resulting from the application of a reduction algorithm and computing the root Hermite factor achieved, we mean the maximum of the corresponding confidence interval.

It is folklore that the output quality of lattice reduction algorithms measured in the root Hermite factor depends mostly on the block size parameter rather than on properties of the input lattice, like the dimension or bit size of the numbers, at least when the lattice dimension and size of the numbers is large enough. A natural approach to comparing the different algorithms would be to fix a number of lattices of certain dimension and bit size and run the different algorithms with varying block size on them. Unfortunately, Slide reduction requires the block size to divide the dimension.[9] To circumvent this we select the dimension of the input lattices depending on the block sizes we want to test, i.e. $n = t \cdot k$, where $k$ is the block size and $t$ is a small integer. This is justified as most lattice attacks involve choosing a suitable sublattice to attack, where such a requirement can easily be taken into account. Since for very small dimensions block reduction performs a little better then in larger dimensions, we need to deal with a trade-off here: on the one hand we need to ensure that the lattice dimension $n$ is large enough, even for small block sizes, so that the result

[9]While it is trivial to generalize Slide reduction to other block sizes, the performance in terms of the achieved output quality of the basis deteriorates somewhat in this case compared to other reduction algorithms [29].

is not biased positively for small block sizes due to the small dimension. On the other hand, if the lattice dimension grows very large we would have to increase the precision of the GSO computation significantly which would result in an artificial slow down and thus limit the data we are able to collect. Our experiments and previous work [16] suggest that the bias for small dimensions weakens sufficiently as soon as the lattice dimension is larger than 140, so for the lattice dimension $n$ we select the smallest multiple $t$ of the block size $k$ such that $t \cdot k \geq 140$.

For each block size we generated 10 different subset sum lattices in dimension $n$ in the sense of [19] and we fix the bit size of the numbers to $10 \cdot n$ following previous work [19,36]. Experimental studies [43] have shown that this notion of random lattices is suitable in this context as lattice reduction behaves similarly on them as on "random" lattices in a mathematically more precise sense [17][10]. Then we ran each of the three reduction algorithms with corresponding block size on each of those lattices. For BKZ and DBKZ we used the same terminating condition: the algorithms terminate when the slope of the shape of the basis does not improve during 5 loop iterations in a row (this is the default terminating condition in `fpLLL`'s BKZ routine with auto abort option set). Finally, for sufficiently large block sizes ($k > 45$), we preprocessed the local blocks with BKZ-$(k/2)$ before calling the SVP oracle, since this has been shown to achieve good asymptotic running time [69] and also seemed a good choice in practice in our experiments.

### 6.2 Results

Figure 2 shows the average output quality including the confidence interval produced by each of the three algorithms in comparison with the prediction based on the Gaussian Heuristic (cf. Equation (4)). It demonstrates that BKZ and DBKZ have comparable performance in terms of output quality and clearly outperform Slide reduction for small block sizes ($< 50$), which confirms previous reports [16]. For some of the small block sizes (e.g. $k = 35$) BKZ seems to perform unexpectedly well in our experiments. To see if this is indeed inherent to the algorithms or a statistical outlier owed to the relatively small number of data points, we ran some more experiments with small block sizes. We report on the results in Appendix A, where we show that the performance of BKZ and DBKZ are actually extremely close for these parameters.

Furthermore, Figure 2 shows that all three algorithms tend towards the prediction given by Equation (4) in larger block sizes, supporting the conjecture, and Slide reduction becomes quite competitive. Even though BKZ still seems to have a slight edge for block size 75, note that the confidence intervals for Slide reduction and BKZ are heavily overlapping here. This is in contrast to the only previous study that involved Slide reduction [16], where Slide reduction was reported to be entirely noncompetitive in practice and thus mainly of theoretical interest.

Figure 3 shows the same data separately for each of the three algorithms including estimated standard deviation. The data does not seem to suggest that one or the other algorithm behaves "nicer" with respect to predictability – the standard deviation ranges between 0.0002 and 0.0004 for all algorithms, but can be as high as 0.00054 (cf. Appendix B). Note that while these numbers might seem small, it affects the base of the exponential that the short vector is measured in, so small changes have a large impact. The standard deviation varies across different block sizes, but there is no evidence that it might converge to smaller values or even 0 in larger block sizes. So we have to assume, that it remains a significant factor for larger block sizes and should be taken into account in cryptanalysis. It is entirely conceivable that the application of a reduction algorithm

[10] In fact, subset sum lattices are extremely similar to the random lattices of [17].

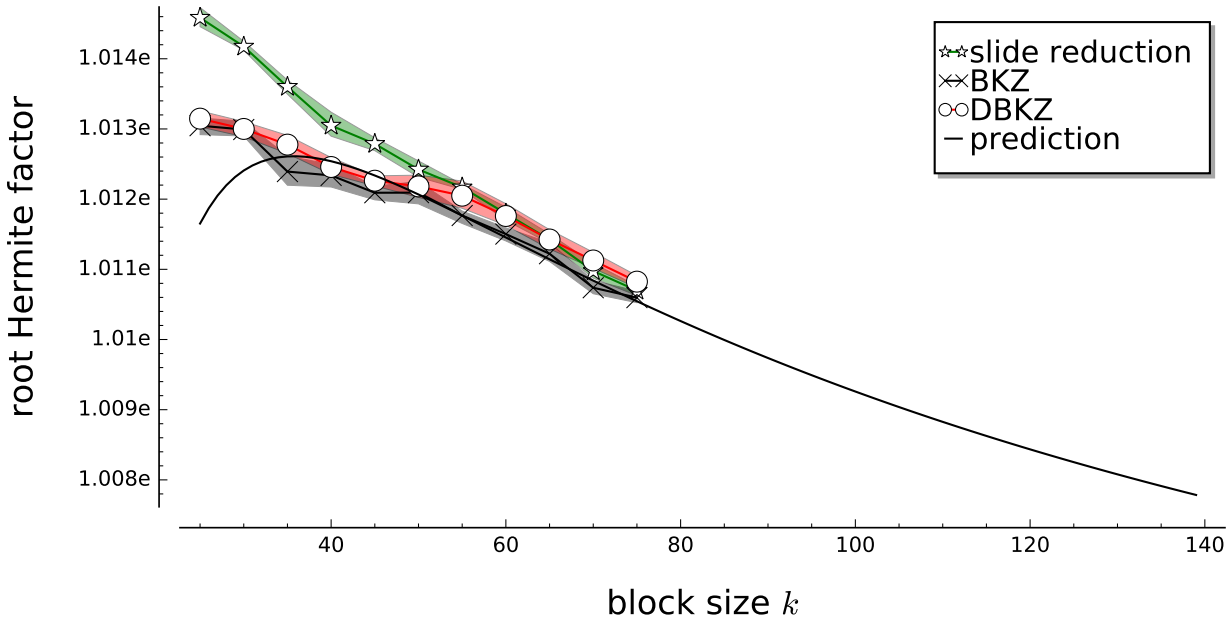


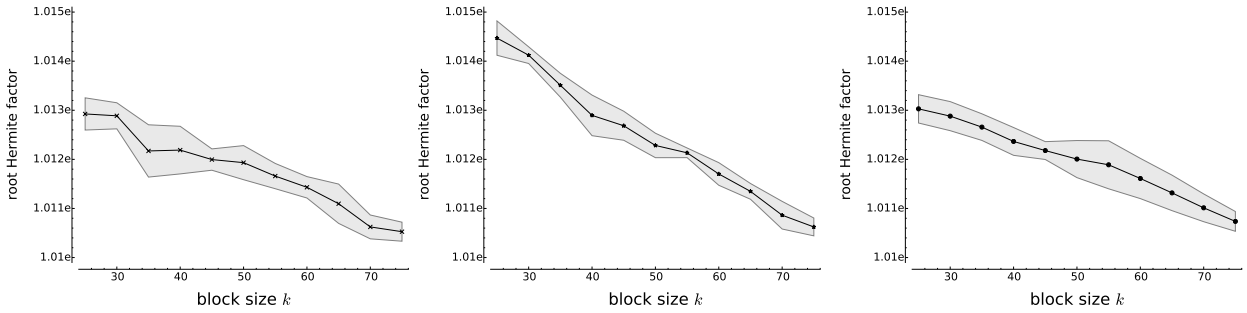

Figure 2: Confidence interval of average root Hermite factor for random bases as computed by different reduction algorithms and the prediction given by Equation (4).

yields a root Hermite factor significantly smaller than the corresponding mean value.

In order to compare the runtime of the algorithms we ran separate experiments, because due to the way we selected the dimension, the data would exhibit a somewhat strange "zigzag" behavior. For each block size $50 \leq k \leq 75$ we generated again 10 random subset sum lattices with dimension $n = 2k$ and the bit size of the numbers was fixed to 1400. Figure 4 shows the average runtime for each of the algorithms and block size in log scale. It shows that the runtime of all three algorithms follows a close to single exponential (in the block size) curve. This supports the intuition that the runtime mainly depends on the complexity of the SVP oracle, since we are using an implementation that preprocesses the local blocks before enumeration with large block size. This has been shown to achieve an almost single exponential complexity (up to logarithmic factors in the exponent) [69].

The data also shows that in terms of runtime, Slide reduction outperforms both, BKZ and DBKZ. But again, with increasing block size the runtime of the different algorithms seem to converge to each other. Combined with the data from Figure 2 this suggests that all three algorithms offer a similar trade-off between runtime and achieved Hermite factor for large block sizes.

This shows that Slide reduction is not only theoretically interesting with its cleaner and tighter analysis of both, output quality and runtime, but also quite competitive in practice. It should be noted that we analyzed Slide reduction as described in [15]. While significant research effort has been spent on improving BKZ, essentially nothing along these lines has been done with regards to Slide reduction. We hope that the results reported here will initiate more research into improvements, both in practice and theory, of Slide reduction.




(a) BKZ (b) Slide Reduction (c) DBKZ

Figure 3: Same as Figure 2 with estimated standard deviation

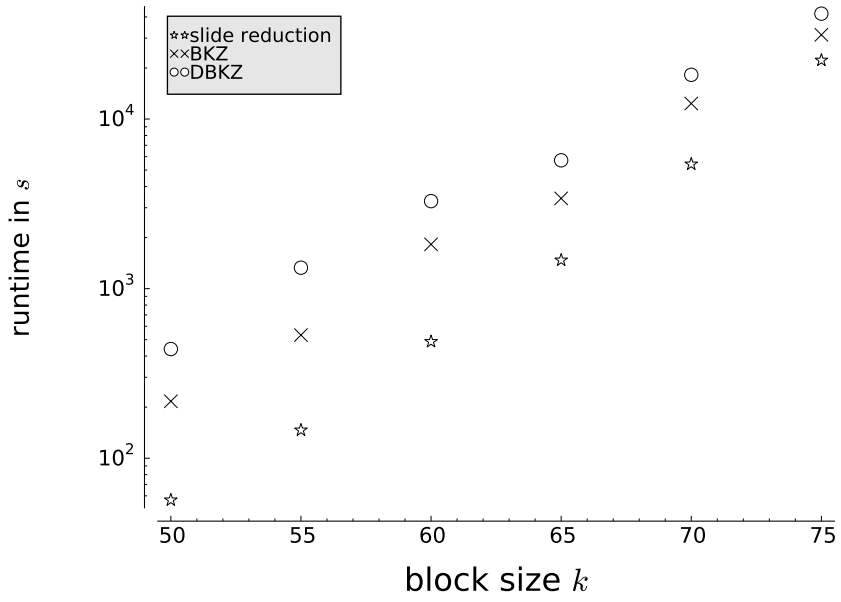



Figure 4: Average runtime in seconds for random bases in dimension $n = 2k$ for different reduction algorithms (in log scale).

# 7 Dual Enumeration

Similar to Slide reduction, DBKZ makes intensive use of dual SVP reduction of projected blocks. The obvious way to achieve this reduction is to compute the dual basis for the projected block, run the primal SVP reduction on it, and finally compute the primal basis of the block. While the transition between primal and dual basis is a polynomial time computation and is thus dominated by the enumeration step, it does involve matrix inversion, which can be quite time consuming in practice. To address this issue, Gama and Nguyen [15] proposed a different strategy. Note that SVP reduction, as performed by enumeration, consists of two steps: (1) the coordinates of a shortest vector in the given basis are computed, and (2) this vector is inserted into the basis. Gama and Nguyen observe that for dual SVP reduction, (2) can be achieved using the coordinates obtained during the dual enumeration by solely operating on the primal basis. Furthermore, note that the enumeration procedure (step (1)) only operates on the GSO of the basis so it is sufficient for (1) to invert the GSO matrices of the projected block, which is considerably easier since they consist of a diagonal and an upper triangular matrix. However, this still incurs a computational overhead of $\Omega(n^3)$.

We now introduce a way to find the coordinates of a shortest vector in the dual lattice without computing the dual basis or dual GSO. As a warm-up, note that the enumeration procedure applied to the dual basis computes all possible coefficients of short vectors in the dual basis. We can enumerate these coefficients $x_i = \langle \mathbf{v}, \mathbf{b}_i \rangle \in \mathbb{Z}$ of the shortest dual vector $\mathbf{v}$ directly and bound the complexity of this approach the following way:

$$x_i = \langle \mathbf{v}, \mathbf{b}_i \rangle = \langle \mathbf{v}, \mathbf{b}_i^* \rangle - \sum_{j<i} \mu_{i,j} \langle \mathbf{v}, \mathbf{b}_j^* \rangle.$$

Setting $x_j^* = \langle \mathbf{v}, \mathbf{b}_j^* \rangle$, which we can compute from $(x_1, \ldots, x_j)$, we get

$$x_i + \sum_{j<i} \mu_{i,j} x_j^* = \langle \mathbf{v}, \mathbf{b}_i^* \rangle$$

and so

$$|x_i + \sum_{j<i} \mu_{i,j} x_j^*| \leq \|\mathbf{v}\| \|\mathbf{b}_i^*\| \leq \frac{\hat{\lambda}_1}{\|\mathbf{d}_i^\dagger\|}$$

which is exactly what one would expect, i.e. it corresponds to the complexity of enumerating all dual lattice points in a parallelepiped. This simple observation suggests that there is an algorithm to compute the coordinates of the shortest dual vector just as efficient as for the shortest primal vector. However, without the dual basis there is no obvious way to evaluate the quality of a solution, i.e. the lengths of a dual vector given only its coordinates in the dual basis. Furthermore, without being able to evaluate partial solution, we can only enumerate in parallelepipeds, while classical enumeration in the primal is able to enumerate lattice points in a sphere by subtracting the length of partial solutions from the enumeration radius. The following Lemma addresses this issue and will result in a dual enumeration algorithm that strongly resembles the primal enumeration routine.

**Lemma 2** *Let* $\mathbf{B}$ *be a lattice basis and* $\mathbf{w}$ *an arbitrary vector in the linear span of* $\mathbf{B}$*. Let* $\mathbf{x}$ *be the coefficient vector expressing* $\mathbf{w}$ *with respect to the dual basis, i.e.,* $x_i = \langle \mathbf{w}, \mathbf{b}_i \rangle$ *for all* $i \leq n$*. Then,*

*for any $k \leq n$, the (uniquely defined) vector $\mathbf{w}^{(k)} \in \text{span}(\mathbf{B}_{[1,k]})$ such that $\langle \mathbf{w}^{(k)}, \mathbf{b}_i \rangle = x_i$ for all $i \leq k$, can be expressed as $\mathbf{w}^{(k)} = \sum_{i \leq k} \alpha_i \mathbf{b}_i^* / \|\mathbf{b}_i^*\|^2$ where*

$$\alpha_i = x_i - \sum_{j<i} \mu_{i,j} \alpha_j. \tag{15}$$

*Proof* The condition $\mathbf{w}^{(k)} \in \text{span}(\mathbf{B}_{[1,k]})$ directly follows from the definition of $\mathbf{w}^{(k)} = \sum_{i \leq k} \alpha_i \mathbf{b}_i^* / \|\mathbf{b}_i^*\|^2$. We need to show that this vector also satisfies the scalar product conditions $\langle \mathbf{w}^{(k)}, \mathbf{b}_i \rangle = x_i$ for all $i \leq k$. Substituting the expression for $\mathbf{w}^{(k)}$ in the scalar product we get

$$\langle \mathbf{w}^{(k)}, \mathbf{b}_i \rangle = \sum_{j \leq k} \alpha_j \frac{\langle \mathbf{b}_j^*, \mathbf{b}_i \rangle}{\|\mathbf{b}_j^*\|^2} = \sum_{j \leq i} \alpha_j \frac{\langle \mathbf{b}_j^*, \mathbf{b}_i \rangle}{\|\mathbf{b}_j^*\|^2} = \alpha_i + \sum_{j<i} \alpha_j \mu_{i,j} = x_i$$

where the last equality follows from the definition of $\alpha_i$. □

This shows that if we enumerate the levels from $k = 1$ to $n$ (note the reverse order as opposed to primal enumeration) we can easily compute $\alpha_k$ from all the given or previously computed quantities in $O(n)$. The length of $\mathbf{w}^{(k)}$ is given by

$$\|\mathbf{w}^{(k)}\|^2 = \sum_{i \leq k} \alpha_i^2 / \|\mathbf{b}_i^*\|^2 = \|\mathbf{w}^{(k-1)}\|^2 + \alpha_k^2 / \|\mathbf{b}_k^*\|^2. \tag{16}$$

To obtain an algorithm that is practically as efficient as primal enumeration, it is necessary to apply the same standard optimizations known as SchnorrEuchner enumeration to the dual enumeration. It is obvious that we can exploit lattice symmetry and dynamic radius updates in the same fashion as in the primal enumeration. The only optimization that is not entirely obvious is enumerating the values for $x_k$ in order of increasing length of the resulting partial solution. However, from Equation (15) and (16) it is clear that we can start by selecting $x_k = \lfloor \sum_{j<k} \mu_{k,j} \alpha_j \rceil$ in order to minimize the first value of $\alpha_k$, and then proceed by alternating around this first value just as in the SchnorrEuchner primal enumeration algorithm.

It is also noteworthy that being able to compute partial solutions even allows us to apply pruning [14] directly. In summary this shows that dual SVP enumeration should be just as efficient as primal enumeration. To illustrate this, Algorithm 2 and 3 show the SchnorrEuchner variant of the two enumeration procedures.[11]

[11] The function `nextX` simply selects the next value for a specific variable in order to alternate correctly around the center of the interval of valid values. We omit details here since it works identical in both algorithms and requires auxiliary variables that would clutter the code unnecessarily.

**Algorithm 2** Dual Enumeration

`procedure` DualEnum($\mu$, $(\|\mathbf{b}_i^*\|^2)_{i\in[n]}$, $A$)
**Input:** The GSO of a lattice $\mu$ and $(\|\mathbf{b}_i^*\|^2)_{i\in[n]}$ and an upper bound $A$ to the squared length of a shortest dual vector
**Output:** The coordinates of a shortest dual vector in the dual basis $\mathbf{D}$

```
1   k ← 1
2   while k ≥ 1
3     α_k ← x_k − Σ_{j<k} μ_{k,j} α_j
4     l_k ← l_{k−1} + α_k^2 / ‖b_k^*‖^2
5     if l_k ≤ A and k = n then
6       s ← x, A ← l_k
7     if l_k ≤ A and k < n then
8       k ← k + 1, x_k ← ⌊Σ_{j<k} μ_{k,j} α_j⌉
9     else
10      k ← k − 1, x_k ← nextX(k)
11  return s
```

**Algorithm 3** Primal Enumeration

`procedure` PrimalEnum($\mu$, $(\|\mathbf{b}_i^*\|^2)_{i\in[n]}$, $A$)
**Input:** The GSO of a lattice $\mu$ and $(\|\mathbf{b}_i^*\|^2)_{i\in[n]}$ and an upper bound $A$ to the squared length of a shortest vector
**Output:** The coordinates of a shortest vector in the basis $\mathbf{B}$

```
1   k ← n
2   while k ≤ n
3     α_k ← x_k + Σ_{j>k} μ_{j,k} x_j
4     l_k ← l_{k+1} + α_k^2 ‖b_k^*‖^2
5     if l_k ≤ A and k = 1 then
6       s ← x, A ← l_k
7     if l_k ≤ A and k > 1 then
8       k ← k − 1, x_k ← ⌊− Σ_{j>k} μ_{j,k} x_j⌉
9     else
10      k ← k + 1, x_k ← nextX(k)
11  return s
```

**Implementation Notes** To give some experimental evidence that the dual enumeration is just as efficient as primal enumeration, we implemented it in `fpLLL`.[12] Note that Algorithm 2 can be easily added to an implementation of Algorithm 3 by using special cases in data accesses and a few operations. Furthermore, in order to avoid the division in line 4 we precomputed the values $1/\|\mathbf{b}_k^*\|^2$ for all $k$. We compared the implementation with the primal enumeration on 10 random bases (in the same sense as in Section 6) in dimension $35 \leq n \leq 50$. As expected, the rate of enumeration was close to equal in both cases – around $3.2 \cdot 10^7$ nodes per second (cf. Table1), which corresponds to slightly more than 100 cycles per node on our 3.4 Ghz test machine. The slight discrepancies (and the low rate for $n = 35$) can be explained by the variable number of nodes that were enumerated and thus certain setup costs are amortized over a different number of nodes.

| $n$ | 35 | 40 | 45 | 50 |
|---|---|---|---|---|
| primal | 2.73 | 3.16 | 3.13 | 3.17 |
| dual | 2.77 | 3.19 | 3.18 | 3.27 |

Table 1: Rate of enumeration (in $10^7$ nodes per $s$) in primal and dual enumeration

**Inserting the Shortest Dual Vector** For completeness we briefly review how to insert the shortest dual vector into the dual basis using its coordinates in the dual basis and operating solely on the primal basis. Let $\mathbf{v}$ be a shortest dual vector. Its coefficients in the dual basis are given by $\mathbf{x} = \mathbf{B}^T\mathbf{v} \in \mathbb{Z}^n$. Our goal is to transform $\mathbf{B}$ into a basis $\bar{\mathbf{B}}$ of the same lattice such that $\bar{\mathbf{B}}^T\mathbf{v} = \mathbf{e}_n$,

[12] At the point of publication of this work, a modified version of this implementation is now included in the main branch of `fplll`.

where $\mathbf{e}_n$ is the $n$-th vector of the standard basis (which is equivalent to $\mathbf{v}$ being the $n$-th vector of the dual of $\bar{\mathbf{B}}$). This can be achieved by computing a unimodular matrix $\mathbf{U}$ such that $\mathbf{Ux} = \mathbf{e}_n$, for example by computing the GCD of the non zero coefficients of $\mathbf{x}$ (which must be 1 since $\mathbf{v}$ is a shortest dual vector). Then we have $\mathbf{e}_n = \mathbf{Ux} = \mathbf{UB}^T\mathbf{v}$ and so with $\bar{\mathbf{B}} = \mathbf{BU}^T$, we obtain $\mathbf{e}_n = \bar{\mathbf{B}}^T\mathbf{v}$ as desired. Note that we do not need to know $\mathbf{v}$ in order to achieve this – knowing $\mathbf{x}$ is sufficient.

It is not immediately clear that the size of the numbers during the execution of this step is polynomially bounded. However, Gama and Nguyen [15] show that it is if using the right strategy for the GCD computation and we will skip this detail here.

# 8 Conclusion and Future Work

While our experimental study of lattice reduction confirms that the *average* root Hermite factor achieved by lattice reduction is indeed, as conjectured, given by Equation (4), the standard deviation is large enough that it is conceivable that a *single* instance finds a much shorter vector. This means that cryptanalytic estimates should take this into account. Unfortunately, it is at this point unclear how to apply our results to cryptanalysis. Before being able to make meaningful security estimates, there are two issues to solve.

On the one hand, since we usually want to estimate the attack complexity for instances much too hard to solve in practice, we need a way to predict the standard deviation in a similar way as the average root Hermite factor. Given our results (cf. Figure 3), we believe it might be reasonable to assume that the standard deviation is upper bounded by a constant, e.g. $\sigma = 0.0006$, but more work is needed to verify this assumption.

Once we can predict (in a conservative and meaningful way) the mean value and standard deviation of the underlying distribution of the root Hermite factor achieved by lattice reduction, we need to find a way to translate this to a root Hermite factor that is unlikely to be achieved by lattice reduction. Since we do not know the underlying distribution, bounds like Chebyshev's inequality spring to mind. Unfortunately, bounds that make no or very weak assumptions on the underlying distribution are generally very poor and thus would lead to overly conservative estimates. For example, Equation (4) estimates the mean root Hermite factor achieved by lattice reduction with block size $k = 100$ to be $\delta_{100} = 1.00926$. If we use Chebyshev's inequality to estimate $\bar{\delta}_{100}$ such that an attacker running BKZ-100 (which is by no means unrealistic) has a probability of less than 1% (which is actually still quite large) to achieve a root Hermite factor less than $\bar{\delta}_{100}$, we would need to set $\bar{\delta}_{100} = 1.00926 - 10 \cdot \sigma = 1.00326$. This is clearly unrealistic – a root Hermite factor of 1.005 is still believed to be completely out of reach today and our results give no reason to believe otherwise.

It is clear that we need to learn more about the underlying distribution in order to aid parameter selection. For example, using more data one could try to verify experimentally if the distribution follows a (possibly truncated) Gaussian as already suspected in [16] for small block sizes, which would allow for much tighter bounds and meaningful estimates. A brief inspection of our data suggests that this might be true even for larger block sizes, but 10 data points per experiment is not sufficient to allow for any further conclusions about the distribution. In any case, we believe our results show that simply relying on the average of a handful of data points is not very meaningful and we hope that this work can serve as a starting point for more sophisticated approaches to selecting parameters secure against attacks involving lattice reduction.

With our new dual enumeration algorithm we provide another tool to practically examine different reduction algorithms. This should facilitate experimental research into reduction algorithms that make use of dual SVP reduction, like variants of Slide reduction. Future lines of research could explore if, for example, the block Rankin reduction algorithm of [28] can be efficiently implemented by using it to apply the densest sublattice algorithm of [10] to the dual lattice. This could be used to achieve potentially stronger notions of reduction with better output quality.

## Acknowledgment

We thank Arnold Neumaier for completing the analysis of the convergence of DBKZ. Furthermore, we thank Arnold and the anonymous reviewers of the Eurocrypt 2016 committee for many helpful comments on a previous version of this work. We also thank Florian Göpfert for helpful discussions in the early stages of the development of the dual enumeration algorithm.

## References

[1] M. Ajtai. Generating hard instances of lattice problems. *Complexity of Computations and Proofs, Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

[2] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of STOC '97*, pages 284–293. ACM, May 1997.

[3] A. Akhavi and D. Stehlé. Speeding-up lattice reduction with random projections (extended abstract). In E. S. Laber, C. F. Bornstein, L. T. Nogueira, and L. Faria, editors, *LATIN*, volume 4957 of *Lecture Notes in Computer Science*, pages 293–305. Springer, 2008.

[4] M. Albrecht, D. Cadé, X. Pujol, and D. Stehlé. fplll-4.0, a floating-point LLL implementation. Available at `http://perso.ens-lyon.fr/damien.stehle`.

[5] M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015. `http://eprint.iacr.org/`.

[6] A. Bachem and R. Kannan. Lattices and basis reduction algorithm. Technical Report 84-006, Mathematisches Institut, Universität zu Köln, 1984.

[7] Y. Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD thesis, ENS, Paris, 2013. Thse de doctorat dirige par Nguyen, Phong-Quang Informatique Paris 7 2013.

[8] Y. Chen and P. Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.

[9] C. Coupé, P. Nguyen, and J. Stern. The effectiveness of lattice attacks against low-exponent RSA. In *Proc. of PKC '99*, number 1560 in LNCS, pages 204–218. Springer, Berlin Germany, Mar. 1999.

[10] D. Dadush and D. Micciancio. Algorithms for the densest sub-lattice problem. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 1103–1122. SIAM, 2013.

[11] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44:463–471, 1985.

[12] N. Gama, N. Howgrave-Graham, H. Koy, and P. Nguyen. Rankin's constant and blockwise lattice reduction. In *Advances in Cryptology – Proceedings of CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 112–130. Springer, Aug. 2006.

[13] N. Gama, N. Howgrave-Graham, and P. Nguyen. Symplectic lattice reduction and NTRU. In *Proceedings of Eurocrypt*, volume 4004 of *LNCS*, pages 233–253. Springer, May 2006.

[14] N. Gama, P. Nguyen, and O. Regev. Lattice enumeration using extreme pruning. In H. Gilbert, editor, *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 257–278. Springer Berlin / Heidelberg, 2010.

[15] N. Gama and P. Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *Proceedings of STOC*, pages 207–216. ACM, May 2008.

[16] N. Gama and P. Q. Nguyen. Predicting lattice reduction. In *Proceedings of Eurocrypt*, volume 4965 of *LNCS*, pages 31–51. Springer, 2008.

[17] D. Goldstein and A. Mayer. On the equidistribution of Hecke points. *Forum Mathematicum*, 15(2):165 – 189, 2003.

[18] G. Hanrot, X. Pujol, and D. Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 447–464, 2011.

[19] G. Hanrot and D. Stehlé. Improved analysis of Kannan's shortest lattice vector algorithm. In *Proceedings of Crypto* [22], pages 170–186.

[20] B. Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoretical Computer Science*, 41(2–3):125–139, Dec. 1985.

[21] N. Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *Proceedings of Crypto* [22], pages 150–169.

[22] IACR. *CRYPTO 2007*, volume 4622 of *LNCS*. Springer, Aug. 2007.

[23] A. Joux and J. Stern. Lattice reduction: A toolbox for the cryptanalyst. *Journal of Cryptology*, 11(3):161–185, 1998.

[24] R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on theory of computing - STOC '83*, pages 193–206. ACM, Apr. 1983. Journal version in Math. of Operation Research 12(3):415-440, (1987).

[25] S. Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM*, 52(5):789–808, Sept. 2005. Preliminary version in FOCS 2004.

[26] H. Koy and C.-P. Schnorr. Segment LLL-reduction of lattice bases. In J. H. Silverman, editor, *CaLC*, volume 2146 of *Lecture Notes in Computer Science*, pages 67–80. Springer, 2001.

[27] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:513–534, 1982.

[28] J. Li and P. Q. Nguyen. Approximating the densest sublattice from Rankins inequality. *LMS Journal of Computation and Mathematics*, 17:92–111, 2014.

[29] J. Li and W. Wei. Slide reduction, successive minima and several applications. *Bulletin of the Australian Mathematical Society*, 88:390–406, 12 2013.

[30] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In A. Kiayias, editor, *Topics in Cryptology CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011.

[31] V. Lyubashevsky and D. Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *Proceedings of Crypto*, volume 5677 of *LNCS*, pages 577–594. Springer, Aug. 2009.

[32] S. Mehrotra and Z. Li. Segment LLL reduction of lattice bases using modular arithmetic. *Algorithms*, 3(3):224–243, 2010.

[33] D. Micciancio. Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor. *SIAM Journal on Computing*, 34(1):118–169, 2004. Preliminary version in STOC 2002.

[34] D. Micciancio. Inapproximability of the shortest vector problem: Toward a deterministic reduction. *Theory of Computing*, 8(1):487–512, 2012.

[35] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measure. *SIAM Journal on Computing*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

[36] D. Micciancio and M. Walter. Fast lattice point enumeration with minimal overhead. In P. Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 276–294. SIAM, 2015.

[37] D. Micciancio and M. Walter. Practical, predictable lattice basis reduction. Cryptology ePrint Archive, Report 2015/1123, 2015. `http://eprint.iacr.org/`.

[38] A. Neumaier. Bounding basis reduction properties. Cryptology ePrint Archive, Report 2016/004, 2016. `http://eprint.iacr.org/`.

[39] P. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto '97. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, Proceedings of the 19th Annual International Cryptology Conference*, volume 1666 of *LNCS*. Springer, Aug. 1999.

[40] P. Nguyen. Hermite's constant and lattice algorithms. In *The LLL Algorithm*, pages 19–69. Springer, 2010.

[41] P. Nguyen. Lattice reduction algorithms: Theory and practice. In K. Paterson, editor, *Advances in Cryptology EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 2–6. Springer Berlin / Heidelberg, 2011.

[42] P. Nguyen and D. Stehlé. Low-dimensional lattice basis reduction revisited. In D. A. Buell, editor, *Algorithmic number theory: 6th international symposium - ANTS-VI*, volume 3076 of *LNCS*, pages 338–357. Springer, June 2004. Journal version in ACM Trans. on Algorithms.

[43] P. Nguyen and D. Stehlé. LLL on the average. In *Proceedings of ANTS VII*, volume 4076 of *LNCS*, pages 238–256. Springer, July 2006.

[44] P. Nguyen and J. Stern. Cryptanalysis of the Ajtai-Dwork cryptosystem. In *Proceedings of CRYPTO '98*, volume 1462 of *LNCS*, pages 223–242. Springer, Aug. 1998.

[45] P. Nguyen and J. Stern. Lattice reduction in cryptology: an update. In *Proceedings of ANTS-IV*, volume 1838 of *LNCS*, pages 85–112. Springer, July 2000.

[46] P. Nguyen and J. Stern. The two faces of lattices in cryptology. In *Proceedings of CaLC '01*, volume 2146 of *LNCS*, pages 146–180. Springer, Mar. 2001.

[47] P. Q. Nguyen and I. E. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Designs, codes and cryptography*, 30(2):201–217, 2003.

[48] P. Q. Nguyen and D. Stehlé. Low-dimensional lattice basis reduction revisited. *ACM Trans. Algorithms*, 5(4):46, oct 2009. Prelim. version in ANTS 2004.

[49] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of STOC*, pages 333–342. ACM, 2009.

[50] T. Plantard and W. Susilo. Recursive lattice reduction. In *SCN*, 2010.

[51] M. Pohst. A modification of the LLL-reduction algorithm. *Journal of Symbolic Computation*, 4(1):123–127, Aug. 1987.

[52] O. Regev. New lattice based cryptographic constructions. *Journal of the ACM*, 51(6):899–942, 2004. Preliminary version in STOC 2003.

[53] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of ACM*, 56(6):34, Sept. 2009. Preliminary version in STOC 2005.

[54] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2–3):201–224, Aug. 1987.

[55] C.-P. Schnorr. A more efficient algorithm for lattice basis reduction. *Journal of Algorithms*, 9(1):47–62, Mar. 1988.

[56] C.-P. Schnorr. Block reduced lattice bases and successive minima. *Combinatorics, Probability and Computing*, 3:507–522, 1994.

[57] C.-P. Schnorr. Lattice reduction by random sampling and birthday methods. In H. Alt and M. Habib, editors, *STACS*, pages 145–156, 2003.

[58] C. P. Schnorr. Fast LLL-type lattice reduction. *Information and Computation*, 204(1):1–25, Jan. 2006.

[59] C. P. Schnorr. Progress on LLL and lattice reduction. In *The LLL Algorithm*, pages 145–178. Springer, 2010.

[60] C.-P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, Aug. 1994. Preliminary version in FCT 1991.

[61] C.-P. Schnorr, M. Fischlin, H. Koy, and A. May. Lattice attacks on GGH cryptosystem. Rump session of Crypto'97, 1997.

[62] C.-P. Schnorr and H. H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Proceedings of EUROCRYPT '95*, volume 921 of *LNCS*, pages 1–12. Springer, May 1995.

[63] M. Seysen. Simultaneous reduction of a lattice basis and its reciprocal basis. *Combinatorica*, 13(3):363–376, 1993.

[64] V. Shoup. NTL: a library for doing number theory. Available at `http://www.shoup.net/ntl/index.html`.

[65] A. Storjohann. Faster algorithms for integer lattice basis reduction. Technical Report 249, Swiss Federal Institute of Technology, ETH-Zurich, Department of Computer Science, Zurich, Switzerland, July 1996.

[66] B. Vallée, M. Girault, and P. Toffin. How to break Okamoto's cryptosystem by reducing lattice bases. In C. G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88, Proceedings of a Workshop on the Theory and Application of Cryptographic Techniques*, volume 330 of *LNCS*, pages 281–291. Springer, May 1988.

[67] B. Vallée and A. Vera. Probabilistic analyses of lattice reduction algorithms. In *The LLL Algorithm*, pages 71–143. Springer, 2010.

[68] J. van de Pol and N. P. Smart. Estimating key sizes for high dimensional lattice based systems. Cryptology ePrint Archive, Report 2013/630, 2013. `http://eprint.iacr.org/`.

[69] M. Walter. Lattice point enumeration on block reduced bases. In A. Lehmann and S. Wolf, editors, *Information Theoretic Security*, volume 9063 of *Lecture Notes in Computer Science*, pages 269–282. Springer International Publishing, 2015.

[70] L. Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

# A Experimental Comparison for Small Block Size

In this section we present some additional experimental results of BKZ and DBKZ for small block sizes. We used the same notion of random lattices as in Section 6, but we fixed the lattice dimension to $n = 140$. Since lattice reduction is more tractable for small block sizes, we generated 20 random lattices and ran BKZ and DBKZ on each of them with block size ranging from $20 \leq k \leq 50$. The average root Hermite factor achieved by the two algorithms is shown in Figure 5. It shows that the two algorithms perform almost equally well in this regime.

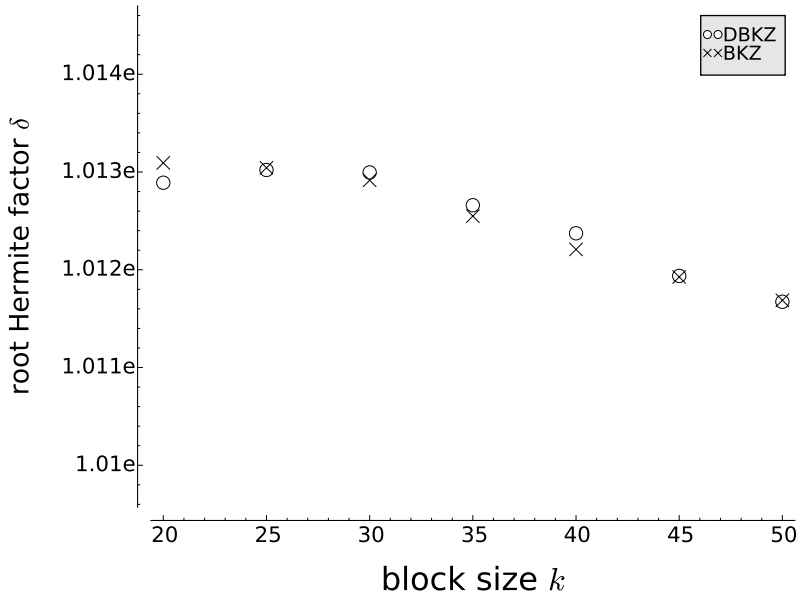



Figure 5: Average root Hermite factor for random lattice in dimension $n = 140$ for BKZ and DBKZ with small block size $k$

# B Experimental Data

| $k$ | $\mu_{[\min]}$ | $\mu_{[\max]}$ | $\sigma_{[\min]}$ | $\sigma_{[\min]}$ |
|---|---|---|---|---|
| BKZ | | | | |
| 25 | 1.0129 | 1.0132 | 0.00014910 | 0.00031542 |
| 30 | 1.0129 | 1.0131 | 0.00013775 | 0.00028044 |
| 35 | 1.0122 | 1.0127 | 0.00030272 | 0.00053605 |
| 40 | 1.0122 | 1.0125 | 0.00020367 | 0.00045143 |
| 45 | 1.0120 | 1.0122 | 0.00011779 | 0.00022000 |
| 50 | 1.0119 | 1.0122 | 0.00019697 | 0.00035792 |
| 55 | 1.0117 | 1.0119 | 0.00012984 | 0.00025347 |
| 60 | 1.0114 | 1.0116 | 0.00009925 | 0.00022226 |
| 65 | 1.0111 | 1.0114 | 0.00012052 | 0.00038869 |
| 70 | 1.0106 | 1.0109 | 0.00012487 | 0.00026133 |
| 75 | 1.0105 | 1.0107 | 0.00009756 | 0.00020439 |
| Slide reduction | | | | |
| 25 | 1.0145 | 1.0148 | 0.00016524 | 0.00034219 |
| 30 | 1.0141 | 1.0142 | 0.00005426 | 0.00017119 |
| 35 | 1.0135 | 1.0137 | 0.00013333 | 0.00022903 |
| 40 | 1.0129 | 1.0133 | 0.00018023 | 0.00041554 |
| 45 | 1.0127 | 1.0129 | 0.00013605 | 0.00027860 |
| 50 | 1.0123 | 1.0125 | 0.00014877 | 0.00026298 |
| 55 | 1.0121 | 1.0122 | 0.00005830 | 0.00009498 |
| 60 | 1.0117 | 1.0119 | 0.00013659 | 0.00022897 |
| 65 | 1.0114 | 1.0115 | 0.00009455 | 0.00017193 |
| 70 | 1.0109 | 1.0111 | 0.00012178 | 0.00023823 |
| 75 | 1.0106 | 1.0108 | 0.00010597 | 0.00019067 |
| Self-Dual BKZ | | | | |
| 25 | 1.0130 | 1.0133 | 0.00012817 | 0.00029479 |
| 30 | 1.0129 | 1.0131 | 0.00017812 | 0.00027150 |
| 35 | 1.0127 | 1.0129 | 0.00016756 | 0.00025963 |
| 40 | 1.0123 | 1.0126 | 0.00013635 | 0.00028876 |
| 45 | 1.0122 | 1.0123 | 0.00010143 | 0.00018625 |
| 50 | 1.0120 | 1.0123 | 0.00018334 | 0.00038216 |
| 55 | 1.0119 | 1.0123 | 0.00026051 | 0.00046222 |
| 60 | 1.0116 | 1.0120 | 0.00018311 | 0.00040919 |
| 65 | 1.0113 | 1.0116 | 0.00014412 | 0.00037256 |
| 70 | 1.0110 | 1.0112 | 0.00013096 | 0.00030097 |
| 75 | 1.0107 | 1.0109 | 0.00011095 | 0.00021169 |

Table 2: Confidence intervals for mean value $\mu$ and standard deviation $\sigma$ of root Hermite factor achieved by lattice reduction with block size $k$