BKZ 2.0: Better Lattice Security Estimates

Yuanmi Chen and Phong Q. Nguyen

[1] ENS, Dept. Informatique, 45 rue d'Ulm, 75005 Paris, France.
http://www.eleves.ens.fr/home/ychen/
[2] INRIA and ENS, Dept. Informatique, 45 rue d'Ulm, 75005 Paris, France.
http://www.di.ens.fr/~pnguyen/

Abstract. The best lattice reduction algorithm known in practice for high dimension is Schnorr-Euchner's BKZ: all security estimates of lattice cryptosystems are based on NTL's old implementation of BKZ. However, recent progress on lattice enumeration suggests that BKZ and its NTL implementation are no longer optimal, but the precise impact on security estimates was unclear. We assess this impact thanks to extensive experiments with BKZ 2.0, the first state-of-the-art implementation of BKZ incorporating recent improvements, such as Gama-Nguyen-Regev pruning. We propose an efficient simulation algorithm to model the behaviour of BKZ in high dimension with high blocksize ≥ 50, which can predict approximately both the output quality and the running time, thereby revising lattice security estimates. For instance, our simulation suggests that the smallest NTRUSign parameter set, which was claimed to provide at least 93-bit security against key-recovery lattice attacks, actually offers at most 65-bit security.

1 Introduction

Lattices are discrete subgroups of $\mathbb{R}^m$. A lattice L is represented by a *basis*, *i.e.* a set of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in $\mathbb{R}^m$ such that L is equal to the set $L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i, x_i \in \mathbb{Z}\}$ of all integer linear combinations of the $\mathbf{b}_i$'s. The integer n is the dimension of L. The goal of *lattice reduction* is to find bases consisting of reasonably short and nearly orthogonal vectors. Lattice reduction algorithms have many applications (see [35]), notably public-key cryptanalysis where they have been used to break special cases of RSA and DSA, among others (see [32] and references therein). There are roughly two types of lattice reduction algorithms:

- *Approximation* algorithms like the celebrated LLL algorithm [22,35], and its blockwise generalizations [41,42,7,8]. Such algorithms find relatively short vectors, but usually not shortest vectors in high dimension.
- *Exact* algorithms to output shortest or nearly shortest vectors. There are space-efficient enumeration algorithms [38,20,6,42,43,10] and exponential-space algorithms [3,36,30,29], the latter being outperformed in practice by the former despite their better asymptotic running time $2^{O(n)}$.

In high dimension, only approximation algorithms can be run, but both types are complementary: approximation algorithms use exact algorithms as subroutines, and exact algorithms use approximation algorithms as preprocessing. In theory, the best approximation algorithm is Gama-Nguyen's reduction [8]. But experiments (such as that of [9], or the cryptanalyses [31,21] of GGH challenges [12]) suggest that the best approximation algorithm known in practice for high dimension is BKZ, published by Schnorr and Euchner in 1994 [42], and implemented in NTL [44]. Like all blockwise algorithms [41,7,8], BKZ has an additional input parameter – the blocksize β – which impacts both the running time and the output quality: BKZ calls many times an enumeration subroutine [38,20,6,42], which looks for nearly-shortest vectors in projected lattices of dimension $\leq \beta$. As β increases, the output basis becomes more and more reduced, but the cost increases significantly: the cost of the enumeration subroutine is typically super-exponential in β, namely $2^{O(\beta^2)}$ polynomial-time operations (see [10]); and experiments [9] show that the number of calls increases sharply with both β and the lattice dimension n: for fixed $\beta \geq 30$, the number of calls looks superpolynomial if not exponential in n. This leads to two typical uses of BKZ:

1. A small blocksize β around 20 in any dimension n, or a medium blocksize β around 30-40 in medium dimension n (say, around 100 at most). Here, BKZ terminates in a reasonable time, and is routinely used to improve the quality of an LLL-reduced basis.
2. A high blocksize $\beta \geq 40$ in high dimension n, to find shorter and shorter lattice vectors. Here, BKZ does not terminate in a reasonable time, and the computation is typically aborted after say, a few hours or days, with the hope that the current basis is good enough for the application: we note that Hanrot *et al.* [14] recently proved worst-case bounds for the output quality of aborted-BKZ, which are only slightly worse than full-BKZ. And one usually speeds up the enumeration subroutine by a pruning technique [42,43,10]: for instance, the implementation of BKZ in NTL proposes Schnorr-Hörner (SH) pruning [43], which adds another input parameter p, whose impact was only clarified in [10]. The largest GGH cryptographic challenges [12] were solved [31,21] using an aborted BKZ of blocksize $\beta = 60$ and SH factor $p = 14$.

One major issue is to assess the output quality of BKZ, especially since lattice algorithms tend to perform better than theoretically expected. The quality is measured by the so-called Hermite factor, as popularized by Gama and Nguyen [9]. In practice, the Hermite factor of all lattice algorithms known is typically exponential in the dimension, namely c^n where c depends on the parameters of the algorithm. The experiments of [9] show that in practice, the Hermite factor of BKZ is typically $c(\beta, n)^n$ where $c(\beta, n)$ quickly converges as n grows to infinity for fixed β. However, the limit values of $c(\beta, n)$ are only known for small values of β (roughly ≤ 30), and theoretical upper bounds [9,14] on $c(\beta, n)$ are significantly higher than experimental values.

All security estimates and proposed parameters (such as recent ones [28,39,23] and NTRU's [18]) of lattice cryptosystems are based on benchmarks of NTL's old implementation of BKZ, but the significance of these estimates is rather debatable. First, these benchmarks were all computed with only usage 1: NTRU [18] "*never observed a noticeable improvement from the pruning procedure, so the pruning procedure was not called*" and used $\beta \leq 25$, while [39,23] use $\beta \leq 30$. This means that such security estimates either assume that BKZ cannot be run with $\beta \geq 30$, or they extrapolate $c(\beta, n)$ for high values of β from low values $\beta \leq 30$. Second, recent progress [10] in enumeration shows that enumeration can now be performed in much higher dimension (*e.g.* $\beta \approx 110$) than previously imagined, but no approximate value of $c(\beta, n)$ is known for large $\beta \geq 50$. And NTL's implementation does not include these recent improvements, and is therefore suboptimal.

Our results. We report the first extensive experiments with high-blocksize BKZ ($\beta \geq 40$) in high dimension. This is made possible by implementing BKZ 2.0, an updated version of BKZ taking into account recent algorithmic improvements. The main modification is the incorporation of the sound pruning technique developed by Gama, Nguyen and Regev [10] at EUROCRYPT '10. The modifications significantly decrease the running time of the enumeration subroutine, without degrading its output quality for appropriate parameters, which allow much bigger blocksizes. BKZ 2.0 outperforms NTL's implementation of BKZ, even with SH pruning [43], which we checked by breaking lattice records such as Darmstadt's lattice challenges [24] or the SVP-challenges [40]: for instance, we find the shortest vector in NTRU [18]'s historical 214-dimensional lattices within $2^{42.62}$ clock cycles, at least 70 times less computation than previously reported [25].

More importantly, our experiments allow us to propose an efficient simulation algorithm to model the execution of BKZ with (arbitrarily) high blocksize ≥ 50, to guess the approximate length of the output vector and the time required: in particular, this algorithm provides the first ever predictions for $c(\beta, n)$ for arbitrarily high values of $\beta \geq 50$. For a given target length, the simulation predicts what is the approximate blocksize β required to obtain such short lattice vectors, and how many enumeration calls will be required approximately. This can be converted into an approximate running time, once we know a good approximation of the cost of enumeration. And we provide such approximations for the best enumeration subroutines known.

Our simulation refines the Gama-Nguyen security estimates [9] on the concrete hardness of lattice problems, which did not take into account pruning, like the security estimates of NTRU [19,16] and those of [23,39]. We illustrate the usefulness of our simulation by revising security estimates. For instance, our simulation suggests that the smallest NTRUSign parameter set, which was claimed to provide at least 93-bit security against key-recovery lattice attacks, actually offers at most 65-bit security. And we use our simulation to provide the first concrete security assessment of the fully-homomorphic encryption challenges [11] recently proposed by Gentry and Halevi. It seems that none of

these challenges offers a very high security level, except the largest one, which seems to offer at most a 100-bit security level.

Roadmap. We start in Sect. 2 with background and notation on lattices. In Sect. 3, we recall the BKZ algorithm. In Sect. 4, we present BKZ 2.0 by describing our modifications to BKZ. In Sect. 5, we briefly report on new lattice records obtained. We present in Sect. 6 a simulation algorithm to predict the performances of BKZ 2.0 with (arbitrarily) high blocksize, which we apply to revise security estimates in Sect. 7. More information can be found in the full version.

2 Preliminaries

We use row representations of matrices (to match lattice software), and use bold fonts to denote vectors: if $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is a matrix, its row vectors are the $\mathbf{b}_i$'s. The Euclidean norm of a vector $\mathbf{v} \in \mathbb{R}^m$ is $\|\mathbf{v}\|$. We denote by $\text{Ball}_n(R)$ the n-dim Euclidean ball of radius R, and by $V_n(R) = R^n \cdot \frac{\pi^{n/2}}{\Gamma(n/2+1)}$ its volume. The n-dim unit sphere is denoted by S^{n-1}. Let L be an n-dim lattice in $\mathbb{R}^m$. Its *volume* $\text{vol}(L)$ is the n-dim volume of the parallelepiped generated by any basis of L.

Orthogonalization. An $n \times m$ basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ can be written uniquely as $B = \mu \cdot D \cdot Q$ where $\mu = (\mu_{i,j})$ is $n \times n$ lower-triangular with unit diagonal, D is $n \times n$ positive diagonal, and Q is $n \times m$ with orthonormal row vectors. Then μD is a lower triangular representation of B (with respect to Q), $B^* = DQ = (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$ is the Gram-Schmidt orthogonalization of the basis, and D is the diagonal matrix formed by the $\|\mathbf{b}_i^*\|$'s. For $1 \leq i \leq n+1$, we denote by π_i the orthogonal projection over $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^\perp$. For $1 \leq j \leq k \leq n$, we denote by $B_{[j,k]}$ the local projected block $\big(\pi_j(\mathbf{b}_j), \pi_j(\mathbf{b}_{j+1}), \ldots, \pi_j(\mathbf{b}_k)\big)$, and by $L_{[j,k]}$ the lattice spanned by $B_{[j,k]}$, whose dimension is $k - j + 1$.

Random Lattices. There is a natural notion of random (real) lattices of given volume, based on Haar measures of classical groups (see [1]). And there is a simple notion of random integer lattices, used in recent experiments: For any integer V, a random n-dim integer lattice of volume V is one chosen uniformly at random among the finitely many n-dim integer lattices of volume V. It was shown in [13] that, as V grows to infinity, the uniform distribution over integer lattices of volume V converges towards the distribution of random (real) lattices of unit volume, once the integer lattice is scaled by $V^{1/n}$. In experiments with random lattices, we mean an n-dim integer lattice chosen uniformly at random with volume a random prime number of bit-length $10n$: for prime volumes, it is trivial to sample from the uniform distribution, using the Hermite normal form. A bit-length $\Theta(n^2)$ would be preferable in theory (in order to apply the result of [13]), but it significantly increases running times, without affecting noticeably experimental results.

Gaussian Heuristic. Given a lattice L and a "nice" set S, the Gaussian Heuristic predicts that the number of points in $S \cap L$ is $\approx \mathrm{vol}(S)/\mathrm{vol}(L)$. In some cases, this heuristic can be proved [1] or refuted [27].

Shortest vector. A *shortest vector* of L has norm $\lambda_1(L) = \min_{\mathbf{v} \in L, \mathbf{v} \neq 0} \|\mathbf{v}\|$, the *first minimum* of L. If the Gaussian heuristic was true for any ball S, we would expect $\lambda_1(L) \approx GH(L)$ where $GH(L) = \mathrm{vol}(L)^{1/n} \cdot V_n(1)^{-1/n}$. Minkowski's theorem shows that $\lambda_1(L) \leq 2GH(L)$ for any lattice L. For random real lattices, $\lambda_1(L)$ is asymptotically equivalent to $GH(L)$ with overwhelming probability (see [1]).

Reduced bases. We recall a few classical reductions. A basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is:

- *size-reduced* if its Gram-Schmidt matrix μ satisfies $|\mu_{i,j}| \leq 1/2$ for $1 \leq j < i \leq n$.
- *LLL-reduced* [22] with factor ε such that $0 < \varepsilon < 1$ if it is size-reduced and its Gram-Schmidt orthogonalization satisfies $\|\mathbf{b}^*_{i+1} + \mu_{i+1,i}\mathbf{b}^*_i\|^2 \geq (1 - \varepsilon)\|\mathbf{b}^*_i\|^2$ for $1 \leq i < n$. If we omit the factor ε, we mean the factor $\varepsilon = 0.01$, which is the usual choice in practice.
- *BKZ-reduced* [41] with blocksize $\beta \geq 2$ and factor ε such that $0 < \varepsilon < 1$ if it is LLL-reduced with factor ε and for each $1 \leq j \leq n$: $\|\mathbf{b}^*_j\| = \lambda_1(L_{[j,k]})$ where $k = \min(j + \beta - 1, n)$.

One is usually interested in minimizing the *Hermite factor* $\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/n}$ (see [9]), which is completely determined by the sequence $\|\mathbf{b}^*_1\|, \ldots, \|\mathbf{b}^*_n\|$. This is because the Hermite factor dictates the performance of the algorithm at solving the most useful lattice problems: see [9] for approx-SVP and unique-SVP, and [28,39,23] for SIS and LWE. It turns out that the Gram-Schmidt coefficients of bases produced by the main reduction algorithms (such as LLL or BKZ) have a certain "typical shape" [9,34], provided that the input basis is sufficiently randomized. To give an idea, the shape is roughly such that $\|\mathbf{b}^*_i\|/\|\mathbf{b}^*_{i+1}\| \approx q$ where q depends on the reduction algorithm, except for the first indexes i. This means that the Hermite factor will typically be of the form c^n where $c \approx \sqrt{q}$.

3 The Blockwise Korkine-Zolotarev (BKZ) Algorithm

3.1 Description

The Blockwise-Korkine-Zolotarev (BKZ) algorithm [42] outputs a BKZ-reduced basis with blocksize $\beta \geq 2$ and reduction factor $\varepsilon > 0$, from an input basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a lattice L. It starts by LLL-reducing the basis B, then iteratively reduces each local block $B_{[j,min(j+\beta-1,n)]}$ for $j = 1$ to n, to make sure that the first vector of each such block is the shortest in the projected lattice. This gives rise to Algorithm 1, which proceeds in such a way that each block is already LLL-reduced before being enumerated: there is an index j, initially

set to 1. At each iteration, BKZ performs an enumeration of the local projected lattice $L_{[j,k]}$ where $k = \min(j + \beta - 1, n)$ to find $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}^n$ such that $\|\pi_j(\sum_{i=j}^{k} v_i \mathbf{b}_i)\| = \lambda_1(L_{[j,k]})$. We let $h = \min(k+1, n)$ be the ending index of the new block in the next iteration:

- If $\|\mathbf{b}_j^*\| > \lambda_1(L_{[j,k]})$, then $\mathbf{b}^{new} = \sum_{i=j}^{k} v_i \mathbf{b}_i$ is inserted between $\mathbf{b}_{j-1}$ and $\mathbf{b}_j$. This means that we no longer have a basis, so LLL is called on the generating set $(\mathbf{b}_1, \ldots, \mathbf{b}_{j-1}, \mathbf{b}^{new}, \mathbf{b}_j, \ldots, \mathbf{b}_h)$, to give rise to a new LLL-reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_h)$.
- Otherwise, LLL is called on the truncated basis $(\mathbf{b}_1, \ldots, \mathbf{b}_h)$.

Thus, at the end of each iteration, the basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is such that $(\mathbf{b}_1, \ldots, \mathbf{b}_h)$ is LLL-reduced. When j reaches n, it is reset to 1, unless no enumeration was successful, in which case the algorithm terminates: the goal of z in Alg. 1 is to count the number of consecutive failed enumerations, to check termination.

Algorithm 1 The Block Korkin-Zolotarev (BKZ) algorithm

Input: A basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, a blocksize $\beta \in \{2, \ldots, n\}$, the Gram-Schmidt triangular matrix μ and $\|\mathbf{b}_1^*\|^2, \ldots, \|\mathbf{b}_n^*\|^2$.
Output: The basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is BKZ-β reduced

$z \leftarrow 0$; $j \leftarrow 0$; LLL($\mathbf{b}_1, \ldots, \mathbf{b}_n, \mu$);*// LLL-reduce the basis, and update μ*
while $z < n - 1$ **do**
  $j \leftarrow (j \mod (n-1)) + 1$; $k \leftarrow \min(j + \beta - 1, n)$; $h \leftarrow \min(k+1, n)$; *// define the local block*
  $\mathbf{v} \leftarrow$Enum($\mu_{[j,k]}, \|\mathbf{b}_j^*\|^2, \ldots, \|\mathbf{b}_k^*\|^2$); *// find $\mathbf{v} = (v_j, \ldots, v_k) \in \mathbb{Z}^{k-j+1} - \mathbf{0}$ s.t. $\|\pi_j(\sum_{i=j}^{k} v_i \mathbf{b}_i)\| = \lambda_1(L_{[j,k]})$*
  if $\mathbf{v} \neq (1, 0, \ldots, 0)$ **then**
    $z \leftarrow 0$; LLL($\mathbf{b}_1, \ldots, \sum_{i=j}^{k} v_i \mathbf{b}_i, \mathbf{b}_j, \ldots, \mathbf{b}_h, \mu$) at stage j; *//insert the new vector in the lattice at the start of the current block, then remove the dependency in the current block, update μ.*
  else
    $z \leftarrow z + 1$; LLL($\mathbf{b}_1, \ldots, \mathbf{b}_h, \mu$) at stage $h - 1$; *// LLL-reduce the next block before enumeration.*
  end if
end while

3.2 Enumeration subroutine

BKZ requires a subroutine to find a shortest vector in a local projected lattice $L_{[j,k]}$: given as input two integers j and k such that $1 \leq j \leq k \leq n$, output $\mathbf{v} = (v_j, \ldots, v_k) \in \mathbb{Z}^{k-j+1}$ such that $\|\pi_j(\sum_{i=j}^{k} v_i \mathbf{b}_i)\| = \lambda_1(L_{[j,k]})$. In practice, as well as in the BKZ article [42], this is implemented by enumeration. One sets $R = \|\mathbf{b}_j^*\|$ as an initial upper bound of $\lambda_1(L_{[j,k]})$. Enumeration

goes through the *enumeration tree* formed by "half" of the vectors in the local projected lattices $L_{[k,k]}, L_{[k-1,k]}, \ldots, L_{[j,k]}$ of norm at most R. The tree has depth $k-j+1$, and for each $d \in \{0, \ldots, k-j+1\}$, the nodes at depth d are 0 and all $\pi_{k-d+1}(\mathbf{u}) \in L_{[k-d+1,k]}$ where $\mathbf{u} = \sum_{i=j}^{k'} u_i \mathbf{b}_i$ with $j \leq k' \leq k$, $u_{k'} > 0$ and $\|\pi_{k-d+1}(\mathbf{u})\| \leq R$. The parent of a node $\mathbf{u} \in L_{[k-d+1,k]}$ at depth d is $\pi_{k+2-d}(\mathbf{u})$ at depth $d-1$. Child nodes are ordered by increasing Euclidean norm. The Schnorr-Euchner algorithm [42] performs a Depth First Search of the tree to output a nonzero leaf of minimal norm, with the following modification: everytime a new (nonzero) leaf is found, one updates the enumeration radius R as the norm of the leaf. The more reduced the basis is, the less nodes in the tree, and the cheaper the enumeration. The running time of the enumeration algorithm is N polynomial-time operations where N is the total number of tree nodes. If the algorithm did not update R, Hanrot and Stehlé [15] noticed that the number of nodes at depth d could be estimated from the Gaussian heuristic as:

$$H_d(R) = \frac{1}{2} \cdot \frac{V_d(R)}{\prod_{i=k-d+1}^{k} \|\mathbf{b}_i^*\|} = \frac{1}{2} \cdot \frac{R^d V_d(1)}{\prod_{i=k-d+1}^{k} \|\mathbf{b}_i^*\|}. \quad (1)$$

Gama *et al.* [10] showed that this heuristic estimate is experimentally very accurate, at least for sufficiently large $k-j+1$ and typical reduced bases. We can therefore heuristically bound the number of nodes at depth d in the actual Schnorr-Euchner algorithm (with update of R) by setting $R = \lambda_1(L_{[j,k]})$ and $R = \|\mathbf{b}_j^*\|$ in Eq. (1). It is shown in [10] that for typical reduced bases, $H_d(R)$ is maximal around the middle depth $d \approx (k-j)/2$, and the remaining $H_d(R)$'s are significantly smaller.

3.3 Analysis

No good upper bound on the complexity of BKZ is known. The best upper bound known for the number of calls (to the enumeration subroutine) is exponential (see [14]). In practice (see [9]), BKZ with $\beta = 20$ is very practical, but the running time significantly increases for $\beta \geq 25$, making any $\beta \geq 40$ too expensive for high-dimensional lattices. In practice, the quality of bases output by BKZ is better than the best theoretical worst-case bounds: according to [9], the Hermite factor for high-dimensional lattices is typically $c(\beta, n)^n$ where $c(\beta, n)$ seems to quickly converge as n grows to infinity, whereas theoretical upper bounds are $c'(\beta)^n$ with $c'(\beta)$ significantly larger than $c(\beta, n)$. For instance, $c(20, n) \approx 1.0128$ for large n. Furthermore, [14] recently showed that if one aborts BKZ after a suitable polynomial number of calls, one can obtain theoretical upper bounds which are only slightly worse than $c'(\beta)^n$.

4 BKZ 2.0

When the blocksize is sufficiently high, namely ≥ 30, it is known [9] that the overall running time of BKZ is dominated by the enumeration subroutine,

which finds a shortest vector in the m-dimensional local projected lattice $L_{[j,k]}$, using a radius R initially set to $\|\mathbf{b}_j^*\|$, where $1 \leq j \leq k \leq n$ and $m = k - j + 1$.

In this section, we describe BKZ 2.0, an updated version of BKZ with four improvements, which we implemented by modifying NTL [44]'s implementation of BKZ [42]. The first improvement is simply an early-abort, which is common practice in cryptanalysis, and is partially supported by the recent theoretical result of [14]: we add a parameter that specifies how many iterations should be performed, *i.e.* we choose the number of oracle calls; this already provides an exponential speedup over BKZ, because the number of calls seems to grow exponentially for fixed $\beta \geq 30$ according to the experiments of [9]. The other three improvements aim at decreasing the running time of the enumeration subroutine: sound pruning [10], preprocessing of local bases, and shorter enumeration radius. Though these improvements may be considered as folklore, we stress that none had been incorporated in BKZ (except that a weaker form of pruning had been designed by Schnorr and Hörner [43], and implemented in NTL [44]), and that implementing them is not trivial.

4.1 Sound Pruning

Pruning speedups enumeration by discarding certain branches, but may not return any vector, or maybe not the shortest one. The idea of pruned enumeration goes back to Schnorr and Euchner [42], and was first analyzed by Schnorr and Hörner [43] in 1995. It was recently revisited by Gama *et al.* [10], who noticed that the analysis of [43] was flawed and that the pruning was not optimal. They showed that a well-chosen high-probability pruning leads to an asymptotical speedup of $2^{m/4}$ over full enumeration, and introduced an extreme pruning technique which gives an asymptotical speedup of $2^{m/2}$ over full enumeration. We incorporated both pruning with non-negligible probability, and extreme pruning using randomization. Formally, pruning replaces each of the $k - j + 1$ inequalities $\|\pi_{k+1-d}(\mathbf{u})\| \leq R$ for $1 \leq d \leq k - j + 1$ by $\|\pi_{k+1-d}(\mathbf{u})\| \leq R_d \cdot R$ where $0 \leq R_1 \leq \cdots \leq R_{k-j+1} = 1$ are $k - j + 1$ real numbers defined by the pruning strategy. For any bounding function $(R_1, \ldots, R_{k-j+1})$, [10] consider the quantities N' and p_{succ} defined by:

- $N' = \sum_{d=1}^{k-j+1} H'_d$ is a heuristic estimate of the total number of nodes in the pruned enumeration tree, where $H'_d = \frac{1}{2} \frac{R^d V_{R_1,\ldots,R_d}}{\prod_{i=k+1-d}^{k} \|\mathbf{b}_i^*\|}$ and $V_{R_1,\ldots,R_d}$ denotes the volume of $C_{R_1,\ldots,R_d} = \left\{(x_1, \ldots, x_d) \in \mathbb{R}^d,\ \forall 1 \leq i \leq d,\ \sum_{l=1}^{i} x_l^2 \leq R_i^2\right\}$.
- $p_{\text{succ}} = p_{\text{succ}}(R_1, \ldots, R_m) = \Pr_{\mathbf{u} \sim S^{m-1}}\left(\forall i \in [1, m],\ \sum_{l=1}^{i} u_l^2 \leq R_i^2\right)$. Let $\mathbf{t} \in L_{[j,k]}$ be a target vector such that $\|\pi_j(\mathbf{t})\| = R$. If the local basis $B_{[j,k]}$ is assumed to be randomized, then p_{succ} is the probability that $\pi_j(\mathbf{t})$ is a leaf of the pruned enumeration tree, under the (idealized) assumption that the distribution of the coordinates of $\pi_j(\mathbf{t})$, when written in the normalized

Gram-Schmidt basis $(\mathbf{b}_j^*/\|\mathbf{b}_j^*\|, \ldots, \mathbf{b}_k^*/\|\mathbf{b}_k^*\|)$ of the local basis $B_{[j,k]}$, look like those of a uniformly distributed vector of norm $\|\pi_j(\mathbf{t})\|$.

We stress that the assumption is only an idealization: in practice, when m is small, for a non-negligible fraction of the local blocks $B_{[j,k]}$, one of the vectors of $B_{[j,k]}$ is a shortest vector of $L_{[j,k]}$, which should have had zero probability. For the application to BKZ, it makes sense to consider various bounding functions of various p_{succ}, say ranging from 1% to 95%, but with a cost N' as small as possible. Based on the methodology of [10], we performed an automated search to generate such bounding functions, for blocksizes β ranging from 35 to 90 by steps of 5, and p_{succ} ranging from 1% to 95%.

It should be noted that BKZ calls the enumeration subroutine on lattices $L_{[j,k]}$ whose dimension $m = k - j + 1$ is not necessarily equal to β. When $j \leq n - \beta + 1$, the dimension m of the block is equal to β, but when $j \geq n - \beta$, the dimension m of the block is strictly less than β. To avoid generating bounding functions for every dimension, we decided in this case to interpolate the bounding function found for β, and checked that interpolating does not affect much p_{succ}. Finally, in order to boost p_{succ}, we added an optional parameter ν, so that BKZ actually performs ν pruned enumerations, each starting with a different random basis of the same local block. This corresponds to the extreme pruning of [10].

4.2 Preprocessing of Local Blocks

The cost of enumeration is strongly influenced by the quality of the local basis, especially as the blocksize increases: the more reduced the local basis, the bigger the volumes of the local projected lattices $L_{[k-d+1,k]}$, and therefore the less nodes in the most populated depths of the enumeration tree. This is folklore, but since BKZ improves regularly the quality of the basis, one might think there is no need to change the local basis before enumeration. However:

- For each enumeration, the local basis is only guaranteed to be LLL-reduced, even though the whole basis may be more than LLL-reduced.
- In high blocksizes, most enumerations are successful: they find a shorter vector than the first block vector. This implies that a local LLL-reduction will be performed to get a basis from a generating set: see Line 6 in Alg. 1. At the next iteration, the enumeration will proceed on a typical LLL-reduced basis, and not something likely to be better reduced.

This suggests that for most enumerations, the local basis is only LLL-reduced, and nothing more, even though other local bases may be better reduced: this was confirmed by experiments.

Hence, we implemented a simple speedup: ensure that the local basis is significantly more reduced than LLL-reduced before each enumeration, but without spending too much time. We used a recursive aborted-BKZ preprocessing to the local basis before enumeration: we performed an automated search to find good parameters depending on β.

4.3 Optimizing the Enumeration Radius

It is folklore that the enumeration cost is also influenced by the choice of the initial radius R, even though this radius is updated during enumeration. Initially, the radius is $R = \|\mathbf{b}_j^*\|$, but if we knew before hand how short would be the output vector, we would choose a lower initial radius R, decreasing the enumeration time. Indeed, the number of nodes at depth d of the enumeration tree (pruned or not) is proportional to R^d. Unfortunately, not much is known (from a theoretical point of view) on how small should be $\lambda_1(L_{[j,k]})$, except general bounds. So we performed experiments to see what was the final norm found by enumeration in practice: Fig. 1 compares the final norm (found by enumeration) to $GH(L_{[j,k]})$, depending on the starting index j of the local block, for one round of BKZ. For the lowest indices j, one sees that the final norm is significantly lower than $GH(L_{[j,k]})$, whereas for the largest indices, it is significantly larger. In the middle, which accounts for most of the enumerations, the ratio between the final norm and the Gaussian heuristic prediction is mostly within 0.95 and 1.05, whereas the ratio between the norm of the first local basis vector and $GH(L_{[j,k]})$ is typically slightly below 1.1. We therefore used the following optimization: for all indexes j except the last 30 ones, we let $R = \min(\sqrt{\gamma}GH(L_{[j,k]}), \|\mathbf{b}_j^*\|)$ instead of $R = \|\mathbf{b}_j^*\|$, where γ is a radius parameter. In practice, we selected $\sqrt{\gamma} = \sqrt{1.1} \approx 1.05$.

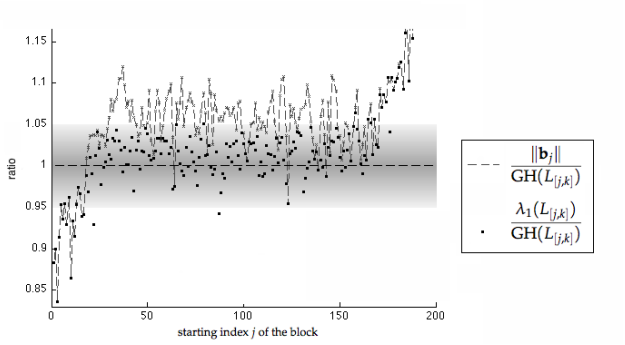

Fig. 1. Comparing $\|\mathbf{b}_j^*\|$, $\lambda_1(L_{[j,k]})$ and $GH(L_{[j,k]})$, for each local block $B_{[j,k]}$.

5 New Lattice Records

Here, we briefly report on experiments using 64-bit Xeon processors to break some lattice records, which suggest that BKZ 2.0 is currently the best lattice reduction algorithm in practice.

5.1 Darmstadt's Lattice Challenge

Darmstadt's lattice challenge [24] started in 2008. For each dimension, the challenge is to find a vector of norm $< q$ in an Ajtai lattice [2], where q de-

pends on the dimension; and try to minimize the norm. Until now, the highest challenge solved was 725: the first solutions to all challenges in dimension 575 to 725 were found by Gama and Nguyen in 2008, using NTL's implementation of BKZ with SH pruning. Shorter solutions have been found since (see the full list [24]), but no challenge of higher dimension had been solved. All solutions were found by reducing appropriate sublattices of much smaller dimension (typically around 150-200), whose existence follows from the structure of Ajtai lattices: we followed the same strategy.

BKZ 2.0 with blocksize 90 (18 pruned-enumerations at 5%) found the first ever solution to challenges 750, 775 and 800, and significantly shorter vectors in all challenges 525 to 725, using in total about 3 core-years, as summarized in Table 1: the first column is the dimension of the challenge, the second one is the dimension of the sublattice we used to find the solution, the third one is the best norm found by BKZ 2.0, the fourth one is the previous best norm found by former algorithms, the fifth one is the ratio between norms, and the sixth one is the Hermite factor of the reduced basis of the sublattice, which turns out to be slightly below $1.01^{\dim}$. The factor $1.01^{\dim}$ was considered to be the state-of-the-art limit in 2008 by Gama and Nguyen [9], which shows the improvement.

Table 1. New Solutions for Darmstadt's lattice challenge [24]

Dim(lattice)	Dim(sublattice)	New norm	Previous norm	Ratio	Hermite factor
800	230	120.054	Unsolved		1.00978^{230}
775	230	112.539	Unsolved		1.00994^{230}
750	220	95.995	Unsolved		1.0976^{220}
725	210	85.726	100.90	0.85	1.00978^{210}
700	200	78.537	86.02	0.91	1.00993^{200}
675	190	72.243	74.78	0.97	1.00997^{190}
650	190	61.935	66.72	0.93	1.00993^{190}
625	180	53.953	59.41	0.91	1.00987^{180}
600	180	45.420	52.01	0.87	1.00976^{180}
575	180	39.153	42.71	0.92	1.00977^{180}
550	180	32.481	38.29	0.85	1.00955^{180}
525	180	29.866	30.74	0.97	1.00990^{180}

5.2 SVP Challenges

The SVP challenge [40] opened in May 2010. The lattices L are random integer lattices of large volume, so that $\lambda_1(L) \approx GH(L)$ with high probability. The challenge is to find a nearly-shortest vector, namely a nonzero lattice vector of norm $\leq 1.05GH(L)$. Using BKZ 2.0 with blocksize 75, 20%-pruning, we were able to solve all challenges from dimension 90 to 112.

6 Predicting BKZ 2.0 by Simulation

We now present an efficient simulation algorithm to predict the performances of BKZ 2.0 with high blocksize $\beta \geq 50$ in high dimension, in terms of running time and output quality. Our simulation is fairly consistent with experiments using several core-years on 64-bit Xeon processors, on random lattices and Darmstadt's lattice challenges. Accordingly, we believe that our simulation can be used to predict approximately what can be achieved using much larger computational power than used in our experiments, thereby leading to more convincing security estimates.

6.1 Description

The goal of our simulation algorithm is to predict the Gram-Schmidt sequence $(\|\mathbf{b}_1^*\|, \|\mathbf{b}_2^*\|, \ldots, \|\mathbf{b}_n^*\|)$ during the execution of BKZ, more precisely at the beginning of every round: a *round* occurs whenever $j = 0$ in Step 3 of Alg. 1, so one round of BKZ costs essentially $n - 1$ enumeration calls. We assume that the input basis is a "random" reduced basis, without special property.

The starting point of our simulation is the intuition, based on Sect. 4.3, that the first minimum of most local blocks looks like that of a random lattice of dimension the blocksize: this phenomenon does not hold in small blocksize ≤ 30 (as noted by Gama and Nguyen [9]), but it becomes more and more true as the blocksize increases, as shown in Fig. 2, where we see that the expectation and the standard deviation of $\frac{\lambda_1(L)}{GH(L)}$ seem to converge to that of a random lattice. Intuitively, this may be explained by a concentration phenomenon: as

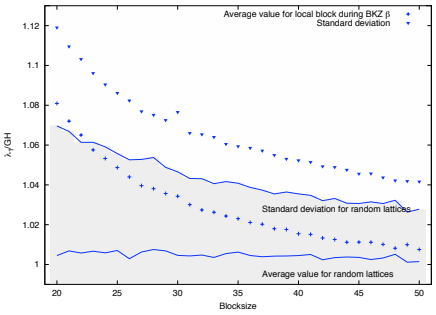

Fig. 2. Comparing $\frac{\lambda_1(L)}{GH(L)}$ for a non-extreme local block during BKZ-β reduction, with a random lattice of dimension β. Expectations with and without standard deviation are given.

the dimension increases, random lattices dominate in the set of lattices, so unless there is a strong reason why a given lattice cannot be random, we may assume that it behaves like a random lattice.

Once we can predict the value of $\lambda_1(L_{[j,k]})$ for each local block, we know that this will be the new value of $\|\mathbf{b}_j^*\|$ by definition of the enumeration subroutine, which allows to deduce the volume of the next local block, and there-

fore iterate the process until the end of the round. This gives rise to our simulation algorithm (see Alg. 2).

Algorithm 2 Simulation of BKZ reduction

Input: The Gram-Schmidt norms, given as $\ell_i = \log(\|\mathbf{b}_i^*\|)$, for $i = 1, \ldots, n$, a blocksize $\beta \in \{45, \ldots, n\}$, and a number N of rounds.
Output: A prediction for the Gram-Schmidt norms $\ell'_i = \log(\|\mathbf{b}_i^*\|), i = 1, \ldots, n$, after N rounds of BKZ reduction.

for $k = 1, \ldots, 45$ **do**
  $r_k \leftarrow$ average $\log(\|\mathbf{b}_k^*\|)$ of an HKZ-reduced random unit-volume 45-dim lattice
end for
for $d = 46, \ldots, \beta$, **do** $c_d \leftarrow \log(GH(\mathbb{Z}^d)) = \log(\frac{\Gamma(d/2+1)^{1/d}}{\pi^{1/2}})$ **end for**
for $j = 1, \ldots, N$ **do**
  $\phi \leftarrow$ true //flag to store whether $L_{[k,n]}$ has changed
  for $k = 1$ to $n - 45$ **do**
    $d \leftarrow \min(\beta, n - k + 1)$ // Dimension of local block
    $f \leftarrow \min(k + \beta, n)$ //End index of local block
    $\log V \leftarrow \sum_{i=1}^{f} \ell_i - \sum_{i=1}^{k-1} \ell'_i$
    if $\phi =$ true **then**
      if $\log V/d + c_d < \ell_k$ **then**
        $\ell'_k \leftarrow \log V/d + c_d$;
        $\phi \leftarrow$ false
      end if
    else
      $\ell'_k \leftarrow \log V/d + c_d$
    end if
  end for
  $\log V \leftarrow \sum_{i=1}^{n} \ell_i - \sum_{i=1}^{n-45} \ell'_i$
  for $k = n - 44$ to n **do**
    $\ell'_k \leftarrow \frac{\log V}{45} + r_{k+45-n}$
  end for
  $\ell_{1,\ldots,n} \leftarrow \ell'_{1,\ldots,n}$
end for

We predict this first minimum $\lambda_1(L_{[j,k]})$ as follows:

- For most indexes j, we choose $GH(L_{[j,k]})$, unless $\|\mathbf{b}_j^*\|$ was already better.
- However, for the last indexes j, namely those inside the last β-dimensional block $L_{[n-\beta+1,n]}$, we do something different: since this last block will be HKZ-reduced at the end of the round, we assume that it behaves like an HKZ-reduced basis of a random lattice of the same volume. Since these averages may be expensive to compute for large β, we apply a simplified rule: we determine the last 45 Gram-Schmidt norms from the average Gram-Schmidt norms (computed experimentally) of an HKZ-reduced basis of a random 45-dim lattice of unit volume, and we compute the first $\beta - 45$ Gram-Schmidt norms using the Gaussian heuristic. But this model

may not work with bases of special structure such as partial reductions of the NTRU Hermite normal form, which is why we only consider random reduced bases as input.

This simulation algorithms allows us to guess the approximate Hermite factor achieved by BKZ 2.0, given an arbitrary blocksize, as summarized in Table 2: for a given dimension n, one should run the simulation algorithm, because the actual blocksize also depends on the dimension. As mentioned in Sect. 2,

Table 2. Approximate required blocksize for high-dimensional BKZ, as predicted by the simulation

Target Hermite Factor	1.01^n	1.009^n	1.008^n	1.007^n	1.006^n	1.005^n
Approximate Blocksize	85	106	133	168	216	286

the Hermite factor dictates the performances at solving lattice problems relevant to cryptography: see [9] for approx-SVP and unique-SVP, and [28,39,23] for SIS and LWE. Obviously, we can only hope for an approximation, since there are well-known variations in the Hermite factor when the input basis is randomized.

The simulation algorithm also gives us an approximate running time, using the number of rounds, provided that we know the cost of the enumeration subroutine: we will discuss these points more precisely later on.

6.2 Consistency with Experiments

It turns out that our simulation matches well with experiments using random lattices and Darmstadt's lattice challenges. First, the prediction of the Gram-Schmidt sequence $(\|\mathbf{b}_1^*\|, \|\mathbf{b}_2^*\|, \ldots, \|\mathbf{b}_n^*\|)$ by our simulation algorithm is fairly accurate for random reduced bases, as shown in Fig. 3 This implies that our

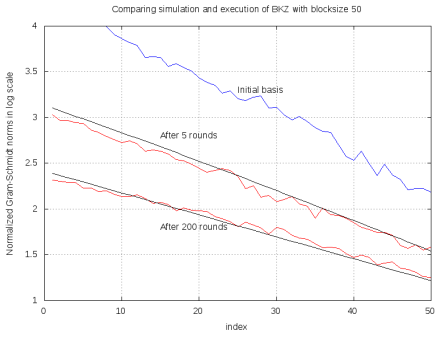

Fig. 3. Predicted vs. actual values of Gram-Schmidt norms during BKZ-50 reduction of a 200-dim random lattice.

simulation algorithm can give a good prediction of the Hermite factor of BKZ

at any given number of rounds, which is confirmed by Fig. 4. Furthermore,

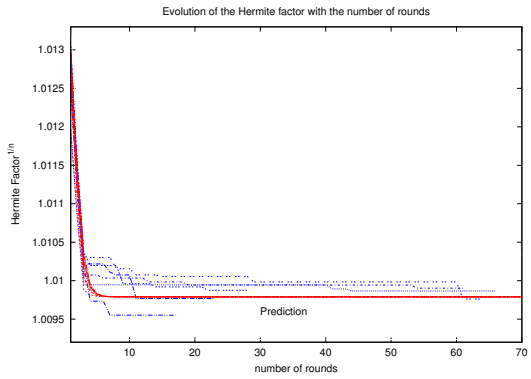

Fig. 4. Evolution and prediction of $(\|\mathbf{b}_1\|/\mathrm{vol}(L)^{1/n})^{1/n}$ during BKZ-90 reduction in dim 180 for Darmstadt's lattice challenges 500–625.

Fig. 4 suggests that a polynomial number of calls seems sufficient to obtain a Hermite factor not very far from that of a full reduction: the main progress seems to occur in the early rounds of BKZ, which justifies the use of aborted-BKZ, which complements the theoretical results of [14].

6.3 Enumeration subroutine

It remains to estimate the cost of the enumeration subroutine, with a radius equal to the Gaussian heuristic. First, we computed upper bounds, by applying extreme pruning on bases reduced with BKZ 2.0, following the search method of [10]: Table 3 gives the approximate cost (in terms of logarithmic number of nodes) of extreme pruning for blocksizes 100-250, using BKZ-75-20% as preprocessing, and radius equal to the Gaussian heuristic. Numbers

Table 3. Upper bound on the cost of the enumeration subroutine, using extreme pruning with aborted-BKZ preprocessing. Cost is given as $\log_2$(number of nodes).

Blocksize	100	110	120	130	140	150	160	170	180	190	200	250
BKZ-75-20%	41.4	47.1	53.1	59.8	66.8	75.2	84.7	94.7	105.8	117.6	129.4	204.1
Simulation of BKZ-90/100/110/120	40.8	45.3	50.3	56.3	63.3	69.4	79.9	89.1	99.1	103.3	111.1	175.2

of nodes can be approximately converted into clock cycles as follows: in the implementation of [10], one node requires about 200 clock cycles for double-precision enumeration, but this figure depends on the dimension, and for high blocksize, we may need higher precision than double precision. For instance, Table 3 says that applying extreme pruning in blocksize 120 would cost at most approximately 2^{53} nodes, which is less than 30 core-years on a 1.86-GHz Xeon, assuming double precision. This is useful to determine parameters for

feasible attacks. However, these upper bounds should not be considered as tight: the performances of enumeration techniques depend on preprocessing, and it is likely that better figures (than Table 3) can be obtained with better preprocessing, including BKZ 2.0 with different parameters. In fact, Table 3 also provides a better upper bound, based on our simulation of BKZ with higher blocksizes 90–120 as a preprocessing. In order to provide security estimates with a good security margin, we need to estimate how much progress can be made. Interestingly, there are limits to enumeration techniques. Nguyen [33] established a lower bound on the number of nodes at each depth of the enumeration tree, assuming that the Gaussian heuristic estimates well the number of nodes (as is usual in analyzing the complexity of enumeration techniques). The lower bounds are based on the Rankin invariants $\gamma_{n,m}(L)$ of a lattice:

$$\gamma_{n,m}(L) = \min_{\substack{S \text{ sublattice of } L \\ \dim S = m}} \left(\frac{\mathrm{vol}(S)}{\mathrm{vol}(L)^{m/n}} \right)^2.$$

In particular, [33] shows that the number of nodes in the middle depth of a full enumeration of a d-dim lattice L with radius $GH(L)$ is $\geq V_{d/2}(1)\sqrt{\gamma_{d,d/2}(L)/V_d(1)}$. For typical lattices L, the Rankin invariant $\gamma_{n,m}(L)$ is heuristically close to the following lower bound on Rankin's constant $\gamma_{n,m}$ (see [7]):

$$\gamma_{n,m} \geq \left(n \frac{\prod_{j=n-m+1}^{n} Z(j)}{\prod_{j=2}^{m} Z(j)} \right)^{\frac{2}{n}} \tag{2}$$

where $Z(j) = \zeta(j)\Gamma(\frac{j}{2})/\pi^{\frac{j}{2}}$ and ζ is Riemann's zeta function: $\zeta(j) = \sum_{p=1}^{\infty} p^{-j}$. These lower bounds are for full enumeration, but they can be adapted to pruning by taking into account the actual speedup of pruning (as analyzed in [10]), which is asymptotically $2^{n/4}$ for high-probability pruning and $2^{n/2}$ for extreme pruning. Table 4 gives the figures obtained with respectively the actual speedup of the so-called linear pruning, and the asymptotical speedup $2^{n/2}$ of extreme pruning. Compared to the upper bounds of Table 3, there is a signifi-

Table 4. Lower bounds on the cost (in log-nodes) of the enumeration subroutine using linear pruning or extreme pruning, following [33,10].

Blocksize	100	120	140	160	180	200	220	240	280	380
Linear pruning	33.6	44.5	56.1	68.2	80.7	93.7	107.0	120.6	148.8	223.5
Extreme pruning	9	15	21.7	28.8	36.4	44.4	52.8	61.5	79.8	129.9

cant gap: the lower bound of linear pruning tells us how much progress could be made if a stronger preprocessing was found for enumeration.

Finally, we note that asymptotically, heuristic variants [36,30,45] of sieve algorithms [3] are faster than pruned enumeration. However, it is unclear how

meaningful it is for security estimates, since these variants require exponential space and are outperformed in practice. And more experiments than [36,30] would be required to evaluate precisely their practical running time. But our model can easily adapt to new progress in the enumeration subroutine, due to Table 2.

7 Revising Security Estimates

Here, we illustrate how our simulation algorithm can be used to obtain arguably better security estimates than previously known.

7.1 NTRU Lattices

In the NTRU cryptosystem [18], recovering the secret key from the public key amounts to finding a shortest vector in high-dimensional lattices of special structure. Because NTRU security estimates are based on benchmarks with BKZ, it is interesting to see the limits of this methodology.

In the original article [18], the smallest parameter set NTRU-107 corresponds to lattices of dimension 214, and it was estimated that key recovery would cost at least 2^{50} elementary operations. The best experimental result to recover the secret key for NTRU-107 by direct lattice reduction (without ad-hoc techniques like [25,26,9] which exploit the special structure of NTRU lattices) is due to May in 1999 [25], who reported one successful experiment using BKZ with SH pruning [43], after 663 hours on a 200-MHz processor, that is $2^{48.76}$ clock cycles. We performed experiments with BKZ 2.0 on 10 random NTRU-107 lattices: We applied LLL and BKZ-20, which takes a few minutes at most; We applied BKZ -65 with 5%-pruning, and checked every 5 minutes if the first basis vector was the shortest vector corresponding to the secret key, in which case we aborted. BKZ 2.0 was successful for each lattice, and the aborted BKZ-65 reduction took less than 2000s on the average, on a 2.83Mhz single core. So the overall running time is less than 40 minutes, that is $2^{42.62}$ clock cycles, which gives a speedup of at least 70, compared to May's experiment, and is significantly lower than 2^{50} elementary operations. Hence, there is an order of magnitude between the initial security estimate of 2^{50} and the actual security level, which is approximately at most 40-bit.

Now, we revisit recent parameters for NTRUSign. In the recent article by Hoffstein *et al.* [17], a summary of the latest parameters for NTRU encryption and signature is given. In particular, the smallest parameter for NTRUsign is $(N, q) = (157, 256)$, which is claimed to provide 80-bit security against all attacks knowns, and 93-bit security against key-recovery lattice attacks. Similarly to [9], we estimate that finding the secret key is essentially as hard as recovering a vector of norm $< q$ in a lattice of dimension $2N = 314$ and volume q^N, which corresponds to a Hermite factor of 1.00886^{2N}. We ran our simulation algorithm for these parameters to guess how many rounds would be required, depending on the blocksize, starting from a BKZ-20 reduced basis

(whose cost is negligible here): about six rounds of BKZ-110 should be sufficient to break NTRUSign-157, which corresponds to roughly 2^{11} enumerations. And according to Table 3, extreme pruning enumeration in blocksize 110 can be done by searching through at most 2^{47} nodes, which corresponds to roughly 2^{54} clock cycles on a typical processor. This suggests that the security level of the smallest NTRUSign parameter against state-of-the-art lattice attacks is at most 65-bit, rather than 93-bit, which is a significant gap.

7.2 Gentry-Halevi's Fully-Homomorphic Encryption Challenges

We now turn to Gentry-Halevi's main Fully-Homomorphic Encryption Challenges [11], for which no concrete security estimate was given. Decrypting a ciphertext amounts to solve a BDD instance, which can be done up to the distance $\min_i \|\mathbf{b}_i\|^*/2$ using Babai's nearest plane algorithm. Targetting a given value of $\min_i \|\mathbf{b}_i\|^*$ can be transformed into a target Hermite factor in the dual lattice. This allows us to estimate the required Hermite factor to solve the BDD instance, based on the approximate distance of the BDD instance and the lattice volume, which is summarized in Table 5.

Table 5. Security Assessment of Gentry-Halevi's main challenges [11]

Dimension n	512	2048	8192	32768
Name	Toy	Small	Medium	Large
Target Hermite factor [9]	1.67^n	1.14^n	1.03^n	1.0081^n
Algorithm expected to decrypt a fresh ciphertext	LLL	LLL	LLL	BKZ with blocksize ≈ 130
Time estimate	30 core-days	$\leq$ 45 core-years	$\leq$ 68582 core-years	$\approx 2^{100}$ clock-cycles

Accordingly, we speculate that decryption for the toy, small and medium challenge can be solved by LLL reduction, which is not straightforward due to the lattice dimension and the gigantic bit-size of the basis (note that there is new theoretical progress [37] on LLL-reduction for large entries). We checked that this was indeed the case for the toy challenge, by performing an actual reduction using a modification of fplll [4]. For the small and medium challenges, we extrapolated running times from truncated challenges, using the fact that our modification of fplll has heuristic running time $O(n^3d^2)$ where d is the bit-size of the lattice volume, where the $\mathcal{O}$ constant depends on the floating-point precision (which increases with the dimension). According to our simulation, breaking the large challenge would require a blocksize ≈ 130 and approximately 60000 rounds (starting from an LLL basis), that is, 2^{31} enumeration calls. Based on Table 3, this enumeration routine would cost at most 2^{60} nodes, so the security offered by the large challenge is at most roughly 100-bit. On the other hand, if ever a stronger preprocessing for enumeration is found, Table 4 suggests that the security level could potentially drop by a factor in the range $2^{10} - 2^{40}$.

Acknowledgements. Part of this work is supported by the Commission of the European Communities through the ICT program under contract ICT-2007-216676 ECRYPT II.

References

1. M. Ajtai. Generating random lattices according to the invariant distribution. Draft of March 2006.
2. M. Ajtai. Generating hard instances of lattice problems. In *Proc. STOC '96*, pages 99–108. ACM, 1996.
3. M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proc. STOC '01*, pages 601–610. ACM, 2001.
4. D. Cadé, X. Pujol, and D. Stehlé. FPLLL library, version 3.0. Sep 2008.
5. L. Devroye. Non-uniform random variate generation, 1986. Available from `http://cg.scs.carleton.ca/~luc/rnbookindex.html`.
6. U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44(170):463–471, 1985.
7. N. Gama, N. Howgrave-Graham, H. Koy, and P. Q. Nguyen. Rankin's constant and blockwise lattice reduction. In *Proc. CRYPTO '06*, volume 4117 of *LNCS*, pages 112–130. Springer, 2006.
8. N. Gama and P. Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *Proc. STOC '08*. ACM, 2008.
9. N. Gama and P. Q. Nguyen. Predicting lattice reduction. In *Proc. EUROCRYPT '08*, volume 4965 of *LNCS*, pages 31–51. Springer, 2008.
10. N. Gama, P. Q. Nguyen, and O. Regev. Lattice enumeration using extreme pruning. In *Proc. EUROCRYPT '10*, volume 6110 of *LNCS*. Springer, 2010.
11. C. Gentry and S. Halevi. Public challenges for fully-homomorphic encryption. Available at `https://researcher.ibm.com/researcher/view_project.php?id=1548`, 2010.
12. O. Goldreich, S. Goldwasser, and S. Halevi. Challenges for the GGH cryptosystem. Available at `http://theory.lcs.mit.edu/~shaih/challenge.html`, 1997.
13. D. Goldstein and A. Mayer. On the equidistribution of Hecke points. *Forum Math.*, 15(2):165–189, 2003.
14. G. Hanrot, X. Pujol, and D. Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In *Proc. CRYPTO '11*, LNCS. Springer, 2011.
15. G. Hanrot and D. Stehlé. Improved analysis of Kannan's shortest lattice vector algorithm. In *Proc. of CRYPTO ' 07*, volume 4622 of *LNCS*. Springer, 2007.
16. P. S. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, and W. Whyte. Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches. In *Proc. ACNS '09*, volume 5536 of *LNCS*, pages 437–455, 2009.
17. J. Hoffstein, N. Howgrave-Graham, J. Pipher, and W. Whyte. Practical lattice-based cryptography: NTRUEncrypt and NTRUSign. 2010. In [35].
18. J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In *Proc. ANTS-III*, volume 1423 of *LNCS*, pages 267–288. Springer, 1998.
19. J. Hoffstein, J. H. Silverman, and W. Whyte. Estimated breaking times for ntru lattices. Technical report, NTRU Cryptosystems, October 2003. Report #012, v2.
20. R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proc. STOC '83*, pages 193–206. ACM, 1983.

21. M. S. Lee and S. G. Hahn. Cryptanalysis of the GGH cryptosystem. *Mathematics in Computer Science*, 3:201–208, 2010.
22. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Ann.*, 261:513–534, 1982.
23. R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. Cryptology ePrint Archive, Report 2010/613. Full version of CT-RSA '11.
24. R. Lindner and M. Rückert. TU Darmstadt lattice challenge. Available at `http://www.latticechallenge.org/`.
25. A. May. Cryptanalysis of NTRU–107. Draft of 1999, available on May's webpage.
26. A. May and J. H. Silverman. Dimension reduction methods for convolution modular lattices. In *Proc. CaLC*, volume 2146 of *LNCS*, pages 110–125. Springer, 2001.
27. J. E. Mazo and A. M. Odlyzko. Lattice points in high dimensional spheres. *Monatsheft Mathematik*, 17:47–61, 1990.
28. D. Micciancio and O. Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, Berlin, 2009.
29. D. Micciancio and P. Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. In *Proc. STOC '10*. ACM, 2010.
30. D. Micciancio and P. Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *Proc. SODA '10*, pages 1468–1480. ACM–SIAM, 2010.
31. P. Q. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto '97. In *Proc. of Crypto '99*, volume 1666 of *LNCS*. Springer, 1999.
32. P. Q. Nguyen. Public-key cryptanalysis. In I. Luengo, editor, *Recent Trends in Cryptography*, volume 477 of *Contemporary Mathematics*. AMS–RSME, 2009.
33. P. Q. Nguyen. Hermite's constant and lattice algorithms. 2010. In [35].
34. P. Q. Nguyen and D. Stehlé. LLL on the average. In *ANTS*, pages 238–256, 2006.
35. P. Q. Nguyen and B. Vallée, editors. *The LLL Algorithm: Survey and Applications*. Information Security and Cryptography. Springer, 2010.
36. P. Q. Nguyen and T. Vidick. Sieve algorithms for the shortest vector problem are practical. *J. of Mathematical Cryptology*, 2(2):181–207, 2008.
37. A. Novocin, D. Stehlé, and G. Villard. An LLL-reduction algorithm with quasi-linear time complexity. In *Proc. STOC '11*. ACM, 2011.
38. M. Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *SIGSAM Bull.*, 15(1):37–44, 1981.
39. M. Rückert and M. Schneider. Estimating the security of lattice-based cryptosystems. Cryptology ePrint Archive, Report 2010/137.
40. M. Schneider and N. Gama. SVP challenge. Available at `http://www.latticechallenge.org/svp-challenge/`.
41. C.-P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2-3):201–224, 1987.
42. C.-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Math. Programming*, 66:181–199, 1994.
43. C.-P. Schnorr and H. H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Proc. of Eurocrypt '95*, volume 921 of *LNCS*. Springer, 1995.
44. V. Shoup. Number Theory C++ Library (NTL) version 5.4.1. Available at `http://www.shoup.net/ntl/`.
45. X. Wang, M. Liu, C. Tian, and J. Bi. Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. Cryptology ePrint Archive, Report 2010/647.