Predicting Lattice Reduction

Nicolas Gama and Phong Q. Nguyen

École normale supérieure/CNRS/INRIA, 45 rue d'Ulm, 75005 Paris, France
`nicolas.gama@ens.fr`
`http://www.di.ens.fr/~pnguyen`

Abstract. Despite their popularity, lattice reduction algorithms remain mysterious cryptanalytical tools. Though it has been widely reported that they behave better than their proved worst-case theoretical bounds, no precise assessment has ever been given. Such an assessment would be very helpful to predict the behaviour of lattice-based attacks, as well as to select keysizes for lattice-based cryptosystems. The goal of this paper is to provide such an assessment, based on extensive experiments performed with the NTL library. The experiments suggest several conjectures on the worst case and the actual behaviour of lattice reduction algorithms. We believe the assessment might also help to design new reduction algorithms overcoming the limitations of current algorithms.

Keywords: Lattice Reduction, BKZ, LLL, DEEP Insertions, Lattice-based cryptosystems.

1 Introduction

Lattices are discrete subgroups of $\mathbb{R}^n$. A lattice L can be represented by a basis, that is, a set of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ in $\mathbb{R}^n$ such that L is equal to the set $L(\mathbf{b}_1, \ldots, \mathbf{b}_d) = \left\{ \sum_{i=1}^{d} x_i \mathbf{b}_i, x_i \in \mathbb{Z} \right\}$ of all integer linear combinations of the $\mathbf{b}_i$'s. The integer d is the dimension of the lattice L. A lattice has infinitely many bases, but some are more useful than others. The goal of *lattice reduction* is to find interesting lattice bases, such as bases consisting of reasonably short and almost orthogonal vectors.

Lattice reduction is one of the few potentially hard problems currently in use in public-key cryptography (see [29,23] for surveys on lattice-based cryptosystems), with the unique property that some lattice-based cryptosystems [3,34,35,33,11] are based on worst-case assumptions. And the problem is well-known for its major applications in public-key cryptanalysis (see [29]): knapsack cryptosystems [32], RSA in special settings [7,5], DSA signatures in special settings [16,26], *etc.* One peculiarity is the existence of very efficient approximation algorithms, which can sometimes solve the exact problem. In practice, the most popular lattice reduction algorithms are: floating-point versions [37,27] of the LLL algorithm [20], the LLL algorithm with deep insertions [37], and the BKZ algorithms [37,38], which are all implemented in the NTL library [39].

Although these algorithms are widely used, their performances remain mysterious in many ways: it is folklore that there is a gap between the theoretical analyses and the experimental performances. In the eighties, the early success of lattice reduction algorithms in cryptanalysis led to the belief that the strongest lattice reduction algorithms behaved as perfect oracles, at least in small dimension. But this belief has shown its limits: many NP-hardness results for lattice problems have appeared in the past ten years (see [23]), and lattice-based attacks rarely work in very high dimension. Ten years after the introduction of the NTRU cryptosystem [15], none of the NTRU challenges has been solved, the smallest one involving a lattice of dimension 334. On the other hand, all five GGH-challenges [12] have been solved [25], except the 400-dimensional one. It is striking to see that the GGH-350 challenge has been solved, while no 334-dimensional NTRU lattice has ever been solved. The behaviour of lattice algorithms is much less understood than that of their factoring and discrete logarithm counterpart. It would be useful to have at least a model (consistent with experiments) for the performances of existing lattice algorithms.

Our Results. We provide a concrete picture of what is achievable today with the best lattice reduction algorithms known in terms of output quality and running time, based on extensive experiments performed with the NTL library during the past year. This sheds new lights on the practical hardness of the main lattice problems, and allows to compare the various computational assumptions (Unique-SVP, Approximate-SVP) used in theoretical lattice-based cryptography [33,11,35,34,3]. For instance, our experiments strongly suggest that Unique-SVP is significantly easier than Approximate-SVP, and that the hardness of Approximate-SVP depends a lot on the structure of the lattice. Our experiments also clarify the gap between the theoretical analyses and the experimental performances of lattice algorithms, and point out several surprising phenomenons on their behaviour. The most important fact is that asymptotically, all the algorithms known seem to only achieve an exponential approximation factor as predicted by theory, but the exponentiation bases turn out to be extremely close to 1, much closer than what theory is able to prove. This seems to nullify the security property of cryptosystems based on the hardness of approximating lattice problems with big polynomial factors, unless such schemes use large parameters. On the other hand, it also makes clear what are the limits of today's algorithms: in very high dimension, today's best algorithms roughly square root the exponential approximation factors of LLL. Our predictions may explain in retrospect why the 350-dimensional GGH lattice has been solved, but not the 334-dimensional NTRU lattices or the 400-dimensional GGH lattice. We believe the assessment might help to design new reduction algorithms overcoming the limitations of current algorithms. As an illustration, we present an alternative attack on the historical NTRU-107 lattices of dimension 214.

Related work. The NTRU company has performed many experiments with BKZ to evaluate the cost of breaking NTRU lattices. However, such experiments only dealt with NTRU lattice bases, which have a very special structure. And their experiments do not lead to any prediction on what can be achieved in general. Our work is in the continuation of that of Nguyen and Stehlé [28] on the average-case of LLL. But the goal of this paper is to provide a much broader picture: [28] only performed experiments with LLL (and not improved algorithms like BKZ which are much more expensive), and focused on the so-called Hermite-SVP problem, without considering cryptographic lattices with special structure.

Road map. The paper is organized as follows. In Section 2, we provide necessary background on lattice reduction. In Section 3, we provide a concrete picture of what lattice reduction algorithms can achieve today. In Section 4, we analyze the experimental running time of lattice reduction algorithms, and point out several unexpected phenomenons. In Section 5, we compare our predictions with former experiments on GGH and NTRU lattices.

2 Background

We refer to [29,23] for a bibliography on lattices.

2.1 Lattices

In this paper, by the term lattice, we mean a discrete subgroup of some $\mathbb{R}^m$. Lattices are all of the form $L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^{n} m_i \mathbf{b}_i \mid m_i \in \mathbb{Z}\}$ where the $\mathbf{b}_i$'s are linearly independent vectors. Such n-tuple of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is called a *basis* of the lattice: a basis will be represented by a row matrix. The *dimension* of a lattice L is the dimension n of the linear span of L. The *volume* of k vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is $\det\left(\langle \mathbf{v}_i, \mathbf{v}_j \rangle\right)^{1/2}_{1 \le i,j \le k}$. The *volume* $\mathrm{vol}(L)$ (or *determinant*) of a lattice L is the volume of any basis of L.

Minima. We denote by $\lambda_i(L)$ the i-th minimum of a lattice L: it is the radius of the smallest zero-centered ball containing at least i linearly independent lattice vectors. The so-called Hermite's constant γ_n of dimension n satisfies Minkowski's second theorem: for any n-dimensional lattice L, and for any $1 \le d \le n$, we have

$$\left(\prod_{i=1}^{d} \lambda_i(L)\right)^{1/d} \le \sqrt{\gamma_n}\,\mathrm{vol}(L)^{1/n}.$$

The exact value of γ_n is only known for $1 \le n \le 8$ and $n = 24$. For other values of n, the best numerical upper bounds known are given in [6]. Asymptotically, Hermite's constant grows linearly in n. Rankin (see [8]) generalized the minima $\lambda_i(L)$ to the smallest subvolumes: $\gamma_{n,m}(L)$ is the minimal value of $\mathrm{vol}(\mathbf{x}_1, \ldots, \mathbf{x}_m)/\mathrm{vol}(L)^{m/n}$ where $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ range over all m linearly independent lattice vectors.

Random lattices. There is a beautiful albeit mathematically sophisticated notion of random lattice, which follows from Haar measures of classical groups. Such measures give rise to a natural probability distribution on the set of lattices: by a random lattice, we mean a lattice picked from this distribution. Random lattices have the following property (see [1] for a proof): with overwhelming probability, the minima of a random n-dimensional lattice L are all asymptotically close to the Gaussian heuristic, that is, for all $1 \leq i \leq n$

$$\frac{\lambda_i(L)}{(\mathrm{vol}L)^{1/n}} \approx \frac{\Gamma(1+n/2)^{1/n}}{\sqrt{\pi}} \approx \sqrt{\frac{n}{2\pi e}}.$$

Many of our experiments use random lattices: by average case, we will mean running the algorithm on a random lattice. To generate random lattices, we use the provable method of [13], like [28].

Random bases. There is unfortunately no standard notion of random bases for a given lattice. By a random basis, we will mean a basis made of rather large vectors, chosen in a heuristic random way (see for instance [12]). Note that it is possible to sample lattice vectors in a sound way, as described by Klein [18] (see a refined analysis in [31,11]). And from any set of linearly independent lattice vectors, one can efficiently derive a basis whose vectors are not much longer (see for instance [2]).

2.2 Lattice problems

The most famous lattice problem is the shortest vector problem (SVP): Given a basis of a lattice L, find a lattice vector whose norm is $\lambda_1(L)$. But SVP has several (easier) variants which are all important for applications:

- Hermite-SVP: Given a lattice L and an approximation factor $\alpha > 0$, find a non-zero lattice vector of norm $\leq \alpha \cdot (\mathrm{vol}L)^{1/n}$. The LLL algorithm [20] and its blockwise generalizations [36,8,10] are designed as polynomial-time Hermite-SVP algorithms. They achieve an approximation factor $(1+\varepsilon)^n$ exponential in the lattice dimension n where $\varepsilon > 0$ depends on the algorithm and its parameters. This exponential factor can actually be made slightly subexponential while keeping the running time polynomial.
- Approx-SVP: Given a lattice L and an approximation factor $\alpha \geq 1$, find a non-zero lattice vector of norm $\leq \alpha \cdot \lambda_1(L)$. Note that it might be difficult to verify a solution to this problem, since $\lambda_1(L)$ may not be known exactly. There are provably secure lattice-based cryptosystems [33,35] based on the worst-case quantum hardness of Approx-SVP with polynomial factor.
- Unique-SVP: Given a lattice L and a gap $\gamma > 1$ such that $\lambda_2(L)/\lambda_1(L) \geq \gamma$, find a shortest vector of L. There are cryptosystems [3,34] based on the worst-case hardness of Unique-SVP with polynomial gap: $n^{1.5}$ for [34] and n^7 for [3].

Any algorithm solving Approx-SVP with factor α also solves Hermite-SVP with factor $\alpha\sqrt{\gamma_n}$. Reciprocally, Lovász [21] showed that any algorithm solving Hermite-SVP with factor α can be used linearly many times to solve Approx-SVP with factor α^2 in polynomial time. There are also reductions [2,24] from the worst-case of Approx-SVP with a certain polynomial factor to the average-case (for a certain class of lattices) of Hermite-SVP with a certain polynomial factor. Any algorithm solving Approx-SVP with factor α also solves Unique-SVP with gap $\geq \alpha$.

We will not discuss the closest vector problem (CVP), which is often used in cryptanalysis. However, in high dimension, the best method known to solve CVP heuristically transforms CVP into Unique-SVP (see for instance the experiments of [25]).

Knapsack lattices. An interesting class of lattices is the Lagarias-Odlyzko lattices [19] introduced to solve the knapsack problem: given n integers $x_1, \ldots, x_n$ uniformly distributed at random in $[1; M]$ and a sum $S = \sum_{i=1}^{n} \epsilon_i x_i$ where $\epsilon_i \in \{0, 1\}$ and $\sum \epsilon_i = \frac{n}{2}$, find all the ϵ_i. The Lagarias-Odlyzko (LO) lattice L [19] has the following property: if the density $d = n/\log_2(M)$ satisfies $d \leq 0.6463\ldots$, then with overwhelming probability, L has a unique shortest vector related to the ϵ_i, and $\lambda_1(L) \approx \sqrt{n/2}$. It has been proved [19] that there exists d_0 such that if $d \leq d_0/n$, then with overwhelming probability over the choice of the x_i's, L has exponential gap, which discloses the ϵ_i by application of LLL.

2.3 Lattice algorithms

When the lattice dimension is sufficiently low, SVP can be solved exactly in practice using exhaustive search, thanks to enumeration techniques [37]. But beyond dimension 100, exhaustive search can be ruled out: only approximation algorithms can be run. Such algorithms try to output lattice bases $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$ with small *approximation factor* $\|\mathbf{b}_1\| / \lambda_1(L)$, or small *Hermite factor* $\|\mathbf{b}_1\| / \mathrm{vol}(L)^{1/n}$. The main approximation algorithms used in practice are the following:

LLL: it is a polynomial-time algorithm [20] which provably achieves (with appropriate reduction parameters) a Hermite factor $\lessapprox (4/3)^{(n-1)/4} \approx 1.075^n$ and an approximation factor $\lessapprox (4/3)^{(n-1)/2} \approx 1.154^n$, where n is the lattice dimension.

DEEP: the LLL algorithm with deep insertions [37] is a variant of LLL with potentially superexponential complexity. It is expected to improve the Hermite factor and the approximation factor of LLL, but no provable upper bound is known (except essentially that of LLL). The implementation of NTL actually depends on a blocksize parameter β: as β increases, one expects to improve the factors, and increase the running time.

BKZ: this is a blockwise generalization of LLL [37] with potentially super-exponential complexity. The BKZ algorithm uses a blocksize parameter β: like DEEP, as β increases, one expects to improve the factors, and increase the running time. Schnorr [36] proved that if BKZ terminates, it achieves an approximation factor $\leq \gamma_\beta^{(n-1)/(\beta-1)}$. By using similar arguments as [36], it is not difficult to prove that it also achieves a Hermite factor $\leq \sqrt{\gamma_\beta}^{1+(n-1)/(\beta-1)}$.

DEEP and BKZ differ from the (theoretical) polynomial-time blockwise generalizations of LLL [36,8,10]: we will see that even the best polynomial-time algorithm known [10] seems to be outperformed in practice by DEEP and BKZ, though their complexity might be superexponential. The recent algorithm of [10] achieves a Hermite factor $\lessapprox \sqrt{\gamma_\beta}^{(n-1)/(\beta-1)}$ and an approximation factor $\lessapprox \gamma_\beta^{(n-\beta)/(\beta-1)}$.

Approximation algorithms exploit the triangular representation of lattice bases, related to orthogonalization techniques. Given a basis $B = [\mathbf{b}_1, ..., \mathbf{b}_n]$, the Gram-Schmidt orthogonalization (GSO) process constructs the unique pair (μ, B^*) of matrices such that $B = \mu B^*$ where μ is lower triangular with unit diagonal and $B^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ has orthogonal row vectors. If we represent the basis B with respect to the orthonormal basis $[\mathbf{b}_1^*/\|\mathbf{b}_1^*\|, \ldots, \mathbf{b}_n^*/\|\mathbf{b}_n^*\|]$, we obtain a triangular matrix whose diagonal coefficients are the $\|\mathbf{b}_i^*\|$'s. Thus, $\mathrm{vol}(B) = \prod_{i=1}^n \|\mathbf{b}_i^*\|$. LLL and BKZ try to limit the decrease of the diagonal coefficients $\|\mathbf{b}_i^*\|$.

It is sometimes useful to look at more than just the quality of the first basis vector $\mathbf{b}_1$. In order to evaluate the global quality of a basis, we define the Gram-Schmidt log (GSL) as the sequence of the logarithms of the $\|\mathbf{b}_i^*\|$: $\mathrm{GSL}(B) = \left(\log\left(\|\mathbf{b}_i^*\| / \mathrm{vol} L^{1/n}\right)\right)_{i=1..n}$. It is folklore that the GSL often looks like a decreasing straight line after running reduction algorithms. Then the average slope η of the GSL can be computed with the least mean squares method: $\eta = 12 \cdot (\sum i \cdot \mathrm{GSL}(B)_i) / ((n+1) \cdot n \cdot (n-1))$. When the GSL looks like a straight line, the Hermite factor H and the average slope η are related by $\log(H)/n \approx -\eta/2$.

3 Experimental quality of lattice reduction algorithms

In this section, we give a concrete picture of what lattice reduction algorithms can achieve today, and we compare it with the best theoretical results known. All our experiments were performed with the NTL 5.4.1 library [39].

First of all, we stress that SVP and its variants should all be considered easy when the lattice dimension is less than 70. Indeed, we will see in Section 4 that exhaustive search techniques [37] can solve SVP within an hour up to dimension 60. But because such techniques have exponential running time, even a 100-dimensional lattice is out of reach.

When the lattice dimension is beyond 100, only approximation algorithms like LLL, DEEP and BKZ can be run, and the goal of this section is to predict what they can exactly achieve. Before giving the experimental results, let us say a few words on the methodology. We have ran experiments on a large number of samples, so that an average behaviour can be reasonably conjectured. For each selected lattice, we ran experiments on at least twenty randomly chosen bases, to make sure that reduction algorithms did not take advantage of special properties of the input basis: the randomization must make sure that the basis vectors are not short. Note that one cannot just consider the Hermite normal form (HNF): for instance, the HNF of NTRU lattices has special properties (half of its vectors are short), which impacts the behaviour of lattice algorithms (see [9]). This means that we will ignore the effect of choosing special input bases: for instance, if one applies LLL on the standard basis of the LO lattice [19], or any permutation of its rows, it can be shown that if the density is d is lower bounded by $d_0 > 0$, then the first vector output by LLL approximates the shortest vector by a subexponential factor $2^{O(\sqrt{n})}$ rather than the general exponential factor $2^{O(n)}$. This phenomenon is due to the structure of orthogonal lattices [29].

Basis randomization allows to transform any deterministic algorithm like LLL or BKZ into a randomized algorithm. Experiments suggest that LLL and BKZ behave like probabilistic SVP-oracles in low dimension (see Fig. 1): no matter which lattice is selected, if the input basis is chosen at random, the algorithm seems to have a non-negligible probability of outputting the shortest vector.

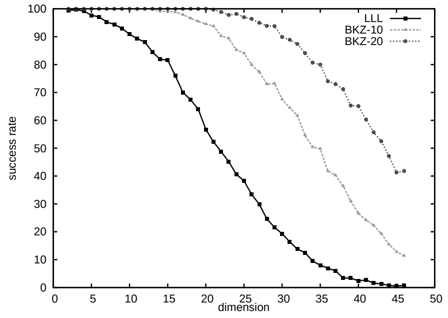

Fig. 1. Experimental probability of recovering the shortest vector, given a random basis of a random lattice, with respect to the dimension.

3.1 Hermite-SVP

The Hermite factor achieved by reduction algorithms seems to be independent of the lattice, unless the lattice has an exceptional structure, in which case the Hermite factor can be smaller than usual (but not higher). By exceptional structure, we mean an unusually small first minimum $\lambda_1(L)$, or more generally, an unusually small Rankin invariant (that is, the existence of a sublattice of unusually small volume). In high dimension, we have never found a class of lattices for which the Hermite factor was substantially higher than for random lattices. We therefore speculate that the worst case matches the average case.

When the lattice has no exceptional structure, the Hermite factor of LLL, DEEP and BKZ seems to be exponential in the lattice dimension: Figure 2 shows the average Hermite factor, with respect to the lattice dimension and the reduction algorithm; and Figure 3 shows the logarithm of Figure 2. The figures show that the Hermite factor is approximately of the form e^{an+b} where n is the lattice dimension and (a, b) seems to only depend on the lattice reduction

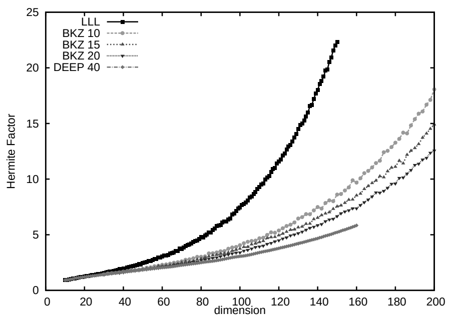

Fig. 2. The Hermite factor of LLL, BKZ and DEEP, depending on the dimension.

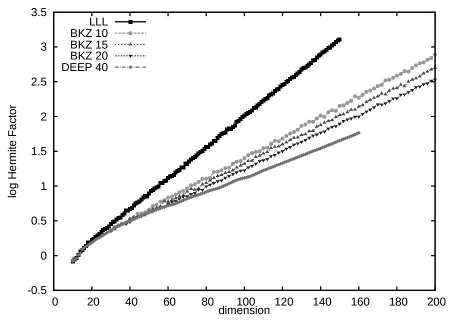

Fig. 3. Logarithm of Figure 2.

	LLL	BKZ-20	BKZ-28	DEEP-50
$c =$ Hermite factor$^{1/n}$	1.0219	1.0128	1.0109	1.011
Best proved upper bound	1.0754	1.0337	1.0282	1.0754
η =average slope GSL	-0.0430	-0.0263	-0.0241	-0.026
Best proved lower bound	-0.1438	-0.0662	-0.0556	-0.1438

Table 1. Average experimental Hermite factor constant of several approximation algorithms on random lattices, and comparison with theoretical upper bounds.

algorithm used. Since we are interested in rough estimations, we simplify e^{an+b} to c^n, and Figure 4 shows that a few samples are enough to have a reasonable approximation of c: indeed, when picking random bases of a given lattice, the distribution looks Gaussian. Figure 5 shows the evolution of c with respect to the lattice dimension and the reduction algorithm; the value c seems to converge as the dimension increases. Table 1 gives the approximate value of c and the corresponding GSL slope η, depending on the algorithm, and compare it with the best theoretical upper bound known. It means that DEEP and BKZ have overall the same behaviour as LLL, except that they give much smaller constants, roughly the square root of that of LLL.

The case of LLL is interesting: it is well-known that the worst-case Hermite factor for LLL is $(4/3)^{(n-1)/4}$, reached by any lattice basis such that all its 2-dimensional projected lattices are critical. However, this corresponds to a worst-case basis, and not to a worst-case lattice. Indeed, when we selected such lattices but chose a random-looking basis, we obtained the same Hermite factor 1.02^n as with random lattices.

One can note that the constant c is always very close to 1, even for LLL, which implies that the Hermite factor is always small, unless the lattice dimension is huge. To give a concrete example, for a 300-dimensional lattice, we obtain roughly $1.0219^{300} \approx 665$ for LLL (which is much smaller than the upper bound $1.0754^{300} \approx 2176069287$) and $1.013^{300} \approx 48$ for BKZ-20 (which is much smaller

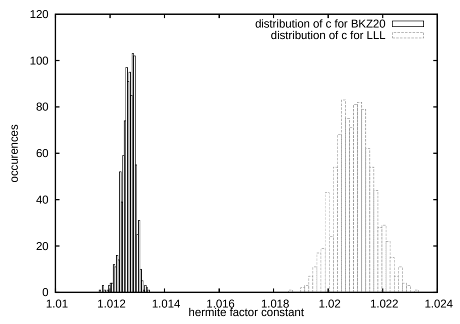

Fig. 4. Distribution of the Hermite factor constant, when picking random bases of a 160-dim lattice.

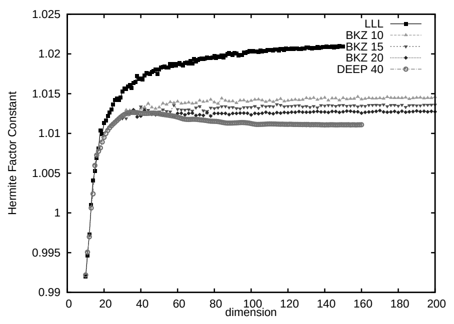

Fig. 5. Convergence of the Hermite factor constant c as the dimension increases.

than the upper bound $1.0337^{300} \approx 20814$). This implies that Hermite-SVP with factor n is easy up to dimension at least 450.

Figure 6 shows the evolution of the Hermite factor constant c for BKZ, as the blocksize increases, and provides two comparisons: one with the best theoretical upper bound known $\approx \sqrt{\gamma_\beta}^{1/(\beta-1)}$, using the best numerical upper bounds known on γ_β, and another with a prototype implementation of the best theoretical algorithm known [10], whose theoretical upper bound is $\sqrt{\gamma_\beta}^{1/(\beta-1)}$. We see that both BKZ and slide reduction [10] perform clearly much better than the theoretical upper bound, but BKZ seems better: slide reduction can be run with a much higher blocksize than BKZ, but even then, the constants seem a bit worse. The size of the gap between theory and practice is hard to explain: we do not have a good model for the distribution of the β-dimensional projected lattices used by BKZ; we only know that it does not correspond numerically to the distribution of a random lattice of dimension β. Figure 7 compares the Hermite factor constant c achieved by BKZ and DEEP, as the blocksize increases. It is normal that the constant achieved by BKZ is lower than DEEP for a fixed blocksize, since BKZ-reduced bases are also necessarily deep-reduced. But the comparison is important, because we will see in Section 4 that one can run DEEP on much bigger blocksize than BKZ, especially for high-dimensional lattices. This opens the possibility that DEEP might outperform BKZ for high-dimensional lattices. Figures 6 and 7 suggest that the best reduction algorithms known can achieve a Hermite factor of roughly 1.01^n in high dimension, but not much lower than that, since BKZ with very high blocksize is not realistic. For instance, a Hermite factor of 1.005^n in dimension 500 looks totally out of reach, unless the lattice has a truly exceptional structure.

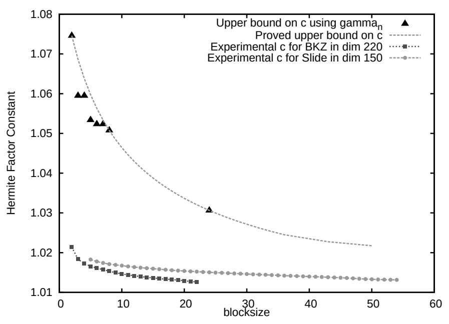

Fig. 6. Average value of the Hermite factor constant c for BKZ in high dimension, depending on the blocksize. Comparison with the best theoretical upper bound and with [10].

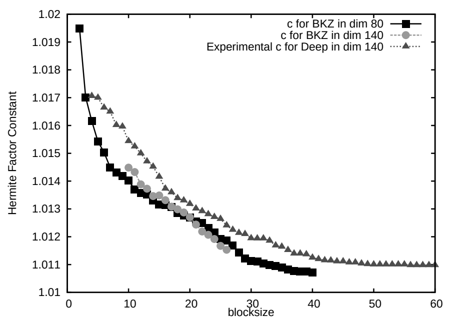

Fig. 7. Comparing the Hermite factor constant c for DEEP in high dimension and BKZ in dimension 80, depending on the blocksize.

3.2 Approx-SVP

As mentioned in Section 2, if we can solve Hermite-SVP with factor c^n in the worst case, then we can solve Approx-SVP with factor $\leq c^{2n}$. Thus, if we believe the previous experimental results on Hermite-SVP, we already expect the best reduction algorithms to solve in practice Approx-SVP with factor roughly $1.01^{2n} \approx 1.02^n$ in the worst case. More precisely, we can square all the values of Table 1 and Figures 6 and 7 to upper bound the approximation factor which can be achieved in practice. This means that Approx-SVP with factor n should be easy up to dimension at least 250, even in the worst case.

Surprisingly, we will see that one can often expect a constant much smaller than 1.02 in practice, depending on the type of lattices. First of all, as noticed in [28], the Hermite factor for random lattices is an upper bound for the approximation factor. More precisely, we know that for a random lattice, $\lambda_1(L)/\mathrm{vol}(L)^{1/n} \approx \frac{\Gamma(1+n/2)^{1/n}}{\sqrt{\pi}} \approx \sqrt{\frac{n}{2\pi e}}$, which means that if the Hermite factor is h, then the approximation factor is $\approx h/\sqrt{\frac{n}{2\pi e}}$. More generally, for any lattice L such that $\lambda_1(L) \geq \mathrm{vol}(L)^{1/n}$, the approximation factor is less than the Hermite factor: this means that on the average, we should achieve 1.01^n rather than 1.02^n. That would imply that Approx-SVP with factor n should be easy on the average up to dimension at least 500.

We have made further experiments to see if the worst case for Approx-SVP corresponds to the square of the Hermite factor, or something smaller. By the previous remark, the worst case can only happen for lattices L such that $\lambda_1(L) \leq \mathrm{vol}(L)^{1/n}$. But if $\lambda_1(L)$ becomes too small compared to $\mathrm{vol}(L)^{1/n}$, reduction algorithms might be able to exploit this exceptional structure to find the shortest vector. After testing various classes of lattices, the worst lattices for Approx-SVP which we have found are the following echelon lattices derived from the classical worst-case analysis of LLL. We call echelon basis a row matrix of the form:

$$\mathrm{Echelon}(\alpha) = \begin{bmatrix} \alpha^{n-1} & 0 & \dots & \dots & 0 \\ \alpha^{n-2}\cdot\sqrt{\alpha^2-1} & \alpha^{n-2} & \ddots & 0 & \vdots \\ 0 & \alpha^{n-3}\cdot\sqrt{\alpha^2-1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \alpha & 0 \\ 0 & \dots & 0 & \sqrt{\alpha^2-1} & 1 \end{bmatrix}, \qquad (1)$$

where $\alpha \in [1; \sqrt{4/3}]$. It is easy to show that the reverse basis $C = (\mathbf{b}_n, \dots, \mathbf{b}_1)$ is HKZ-reduced, and that the successive minima of the echelon lattice L satisfy: $\alpha^{k-1} < \lambda_k(L) \leq \alpha^k$, which allows to precisely estimate $\lambda_1(L)$. We have run the LLL algorithm on many echelon lattices (where the input basis is randomly chosen, not an echelon basis), depending on the value of α. The behaviour of LLL on such lattices is summarized by Figure 8. Two cases can occur:

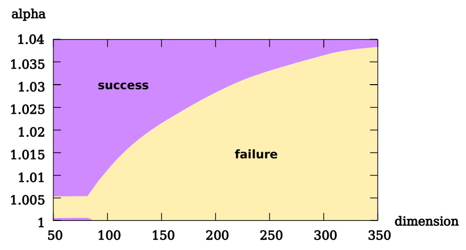

Fig. 8. Behaviour of LLL on echelon lattices, with respect to α and the dimension.

– Either LLL succeeds in finding the shortest vector of the echelon lattice, in which case it actually finds the full HKZ-reduced basis. In particular, this happened whenever $\alpha > 1.043$,
– Either LLL fails to recover the shortest vector. Then the slope of the output GSL and the Hermite factor corresponds to those of random lattices: $c = 1.0219$ and $\eta = -0.043$. This means that the approximation factor of LLL is roughly α^n. Since α can be as high as 1.038 (in dimension 350) in Figure 8, this means that the approximation factor of LLL can be almost as high as the prediction $1.021^{2n} \approx 1.044^n$.

These experiments suggest that the worst case for Approx-SVP is very close to the square of the average Hermite factor for all reduction algorithms known, since this is the case for LLL, and the main difference between LLL and DEEP/BKZ is that they provide better constants. But the experiments also suggest that one needs to go to very high dimension to prevent reduction algorithms to take advantage of the lattice structure of such worst cases.

To summarize, it seems reasonable to assume that current algorithms should achieve in a reasonable time an approximation factor $\leq 1.01^n$ on the average, and $\leq 1.02^n$ in the worst case.

3.3 Unique-SVP

From a theoretical point of view, we know that if one can solve Approx-SVP with factor α in the worst-case, then we can solve Unique-SVP for all gap $\geq \alpha$. The previous section therefore suggests that we should be able to solve any Unique-SVP of gap roughly $\geq 1.02^n$, which corresponds to the square of the Hermite factor. In this section, we present experimental evidence which strongly suggest that Unique-SVP can be solved with a much smaller gap, namely a fraction of the Hermite factor 1.01^n, rather than the square of the Hermite factor. This means that Unique-SVP seems to be significantly easier than Approx-SVP.

The main difficulty with testing the hardness of Unique-SVP is to create lattices for which we precisely know the gap. We therefore performed experiments on various classes of lattices having a unique shortest vector.

Semi-Orthogonal Lattices We first tested lattices for which the shortest vector was in some sense orthogonal to all other lattice vectors. More precisely, we chose lattices L for which the shortest vector $\mathbf{u}$ was such that $L' = L \cap \mathbf{u}^{\perp}$ was equal to the projection of L over $\mathbf{u}^{\perp}$: then $\lambda_2(L) = \lambda_1(L')$ and we chose L' in such a way that $\lambda_1(L')$ could be fixed, so as to select the gap of L. To be concrete, we tested the following two classes of lattices which are parameterized by a given pair (g_1, g_2) of real numbers. The two classes are

$$\begin{bmatrix} g_1 & 0 & \dots & 0 \\ 0 & g_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & g_2 \end{bmatrix} \text{ and } \begin{bmatrix} g_1 & 0 & 0 \dots 0 \\ 0 & M & 0 \dots 0 \\ 0 & r_1 & 1 \ddots \vdots \\ \vdots & \vdots & 0 \ddots 0 \\ 0 & r_{n-1} & 0 \;\; 0 \;\; 1 \end{bmatrix} \text{ where } r_i \in [1; M]$$

where M is a prime number, selected so that $\lambda_2(L) \approx g_2$: to do so, notice that the projection L' can be assumed to be random (see [13]), which gives a formula for $\lambda_1(L')$ depending simply on M.

Notice that the projected lattice L' is a hypercubic lattice for the first class, and a random lattice in the second class. In both cases, $(\mathrm{vol} L)^{1/n} / \lambda_1(L) \approx \lambda_2(L)/\lambda_1(L) \approx g_2/g_1$. The experiments on such lattices have been performed in dimensions 100 to 160, with g_2/g_1 between 2 and 20, and with randomly chosen bases.

For both classes, LLL is able to recover the unique shortest vector as soon as the gap is exponentially large, as shown by Figure 9. More precisely, for the first class, LLL recovers the unique shortest vector with high probability when the gap g_2/g_1 is a fraction of the Hermite factor, as shown by Figure 9 for instance

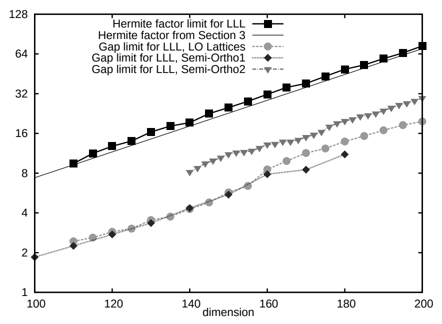

Fig. 9. Gap limits for solving Unique-SVP with LLL, and comparison with the Hermite factor.

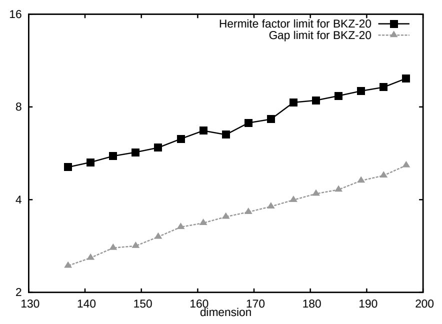

Fig. 10. Same as Figure 9, but with BKZ-20 on LO lattices.

$\geq 0.26 \cdot 1.021^n$ for the first class, and $\geq 0.45 \cdot 1.021^n$ for the second class. The smaller constants in the first class can perhaps be explained by the presence of an unusually orthogonal basis in the projected lattice, which triggers the success of LLL. Again, the behaviour of BKZ is similar to LLL, except that the constants are even smaller: in fact, the constants are so close to 1 that lattice dimensions $<$200 are too small to have good accuracy on the constants. For instance, BKZ-20 finds the shortest vector in dimension 200 in the first class, as soon as the gap is ≥ 2.09, and this limit grows up to 6.4 in dimension 300. This suggests that BKZ-20 retrieves the shortest vector when the gap is $\geq 0.18 \cdot 1.012^n$. Surprisingly, we will see in Section 5 that these very approximate BKZ-20 constants seem consistent with past high-dimensional experiments on the GGH challenges [12].

Knapsack lattices The previous lattices have an exceptional structure compared to a general unique-SVP instance, which might bias the results. This suggests to test other types of lattices, such as the Lagarias-Odlyzko lattices [19]. In order to compare the results with those on semi-orthogonal lattices, we need to estimate the gap of LO lattices. Unfortunately, no provable formula is known for the second minimum of LO lattices. However, the analysis of Nguyen and Stern [30] suggests to heuristically estimate the gap from combinatorial quantities. More precisely, let $N(n,r)$ be the number of vectors in $\mathbb{Z}^n$ or norm $\leq \sqrt{r}$, which can easily be computed numerically. When r becomes large enough that $N(n,r) \gg M$, this hints that $\lambda_2(L) \approx \sqrt{r}$ (see [30]). It can be checked experimentally in low dimension that this heuristic approximation is very precise. As shown in Figures 9 and 10, the minimum gaps for which LLL or BKZ retrieve the shortest vector are once again proportional to the corresponding Hermite factors, that is in $0.25 \cdot 1.021^n$ for LLL and $0.48 \cdot 1.012^n$ for BKZ-20.

4 Running times

In the previous section, we gave experimental estimates on the output quality of reduction algorithms. In this section, we now analyze the running-time growth to see if there are surprising phenomenons, and to guess what can be achieved in a reasonable time. We mainly ran the BKZ routine of NTL with quadratic precision to avoid floating-point issues, so the running times should not be considered as optimal.

4.1 Exhaustive search

In low dimension, SVP can be solved exactly by exhaustive search: in practice, the most efficient method known is Schnorr-Euchner [37]'s enumeration, which is used as a subroutine in BKZ, and which outperforms the theoretical algorithms of Kannan [17] and AKS [4] (even though they have a much better theoretical complexity, see [31]). Given as input a reduced basis (the more reduced the basis, the faster the enumeration), it outputs the shortest vector in $2^{O(n^2)}$ polynomial-time operations. Figure 11 shows the average experimental running time of the enumeration (on a 1.7Ghz 64-bit processor), depending on the quality of the input basis (LLL, BKZ or DEEP). One can see that when the input basis is only LLL-reduced, the running time looks indeed superexponential $2^{O(n^2)}$. We also see that SVP can be solved in dimension 60 within an hour, but the growth of the curve also shows that a 100-dimensional lattice would take at least 35,000 years. A stronger preprocessing will reduce the curve a bit, but it is unlikely to make 100-dimensional lattices within reach.

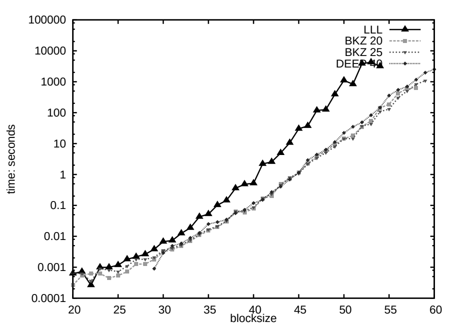

Fig. 11. Running time of the Schnorr-Euchner exhaustive search, depending on the preprocessing.

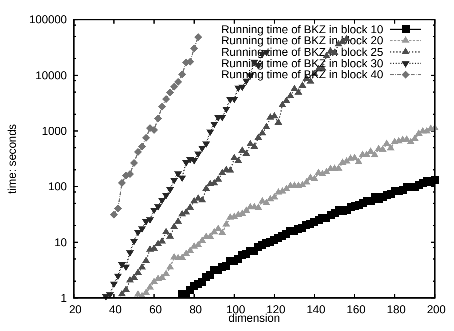

Fig. 12. Running time of BKZ in fixed blocksize.

4.2 BKZ

No good upper bound on the complexity of BKZ and DEEP is known. If β is the blocksize and n is the lattice dimension, the best upper bound is $(n\beta)^n$ polynomial-time operations, which is super-exponential. But this upper bound does not seem tight: it only takes a few seconds to reduce a 100-dimensional lattice with blocksize 20. Since the theoretical analysis is not satisfying, it is very important to assess the experimental running time of BKZ, which is shown in Figures 13 and 12. Obviously, for fixed dimension, the running time of BKZ

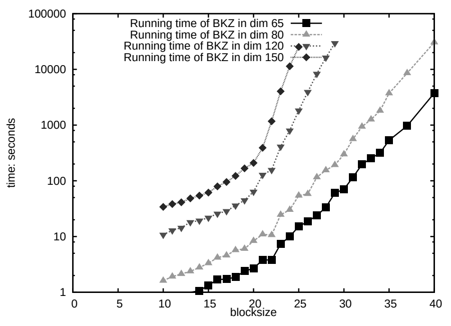

Fig. 13. Running time of BKZ in fixed dimension.

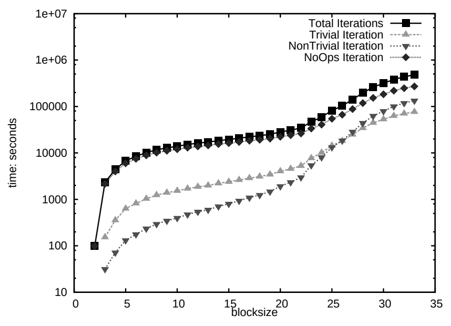

Fig. 14. Number of iterations in BKZ in dimension 100.

increases with the blocksize. But one can observe a brutal increase in the running time around blocksize 20 to 25 in high dimension, and the slope of the increase sharpens with the lattice dimension. We tried to determine the cause of this sudden increase. The increase does not seem to be caused by floating-point inaccuracies, as experiments with higher floating-point precision led to a similar phenomenon: Nor is it caused by the cost of the Schnorr-Euchner enumeration: exhaustive searches typically represent less than 1% of the total reduction time in blocksize 25. In fact, it seems to be caused by a sudden increase in the number of calls to the Schnorr-Euchner enumeration. During a BKZ reduction, each exhaustive search inside a block gives rise to three possibilities:

1. Either the first block basis vector $\mathbf{b}_i^*$ is the shortest lattice vector in the block. Such cases are counted by `NoOps` in NTL.
2. Either the shortest lattice vector in the block is one of the β projected basis vectors. Such cases are counted by `Triv` in NTL.
3. Otherwise, the shortest lattice vector is neither of the β projected basis vectors. Then the algorithm has to do more operations than in the previous two cases. Such cases are counted by `NonTriv` in NTL.

After monitoring (see Figure 14), we observed that the `NoOps` case occurred most of the time, followed by `Triv` reductions and `NonTriv` reductions for blocksizes lower than 25. For higher blocksizes, `NoOps` was still the majority, but `NonTriv` iterations occurred more times than `Triv` iterations.

From Figures 13 and 12, we deduce that blocksizes much higher than 25 are not realistic in very high lattice dimension: the running time seems to be exponential in the dimension when the blocksize is ≥ 25, This is why we estimated the feasibility limit of the Hermite factor to roughly 1.01^n in Section 3, based on Figures 6 and 7: even if we were able to use blocksize 32, we would still not beat 1.01^n.

4.3 DEEP

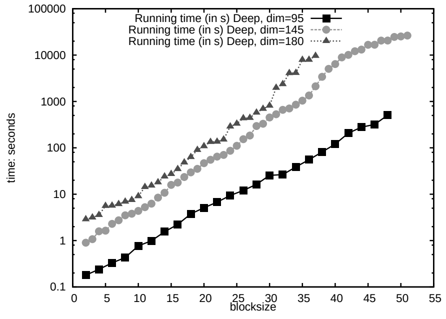

Fig. 15. Running time of DEEP in fixed dimension.

Figure 15 gives the running time of the DEEP algorithm implemented in NTL, depending on the blocksize. Compared to Figure 13, we see that the running time of DEEP is much more regular than BKZ: there is no sharp increase at blocksize 20-25; the running time grows exponentially on a regular basis. Also the slope of the running-time of DEEP (in logarithmic scale) does not increase with the dimension of the lattice. This suggests that DEEP can be run in very high dimension with much higher blocksize than BKZ, which may make DEEP preferable to BKZ. However, Figure 7 showed that even with much higher blocksize, we do not expect to go significantly below the 1.01^n prediction for the Hermite factor.

5 Comparison with former lattice-based attacks

In Section 3, we tried to predict the asymptotical behaviour of the best reduction algorithms known. In this section, we compare our predictions with the largest lattice experiments ever done: surprisingly, our predictions seem consistent with the experiments, and may explain in retrospect why certain lattice attacks worked, but not others.

5.1 The GGH Challenges

In 1999, Nguyen [25] broke four GGH-challenges [12] in dimension 200, 250, 300 and 350, but the 400-dimensional challenge remained unbroken. The attack heuristically transformed a CVP-instance into a Unique-SVP instance, where a heuristic value for the gap of the Unique-SVP instance was known. The Unique-SVP instances arising from GGH-challenges look a bit like the first class of semi-orthogonal lattices: this is because GGH secret bases are slight perturbations of a multiple of the identity matrix.

By extrapolating the experimental results of Section 3, we can make a very rough guess of what should be the gap limit for which the BKZ-20 algorithm would solve the Unique-SVP instance corresponding to the GGH challenge. The results are given in Table 2. Even though the prediction $0.18 \cdot 1.012^n$ is only

Dimension n	200	250	300	350	400
Estimation of the GGH gap	9.7	9.4	9.5	9.4	9.6
Gap estimate for BKZ-20 from Section 3	2.00	3.55	6.44	11.71	21.25
Algorithm used in [25]	BKZ-20	BKZ-20	BKZ-20	pruned-BKZ-60	Not broken

Table 2. Comparing predictions with past experiments on the GGH challenges.

a rough estimate, the difference of magnitude shows that in retrospect, it was not a surprise that Nguyen [25] solved the GGH-challenges with BKZ-20 in dimension 200, 250 and 300. In dimension 350, the prediction is a bit worse, which is consistent with the fact that BKZ-20 failed: Nguyen [25] had to use a pruned BKZ-reduction to solve the GGH-350 challenge. In dimension 400, the prediction is much worse than the expected gap, and it is therefore not a surprise that GGH-400 has not been solved. It seems that we would need much stronger reduction algorithms to solve GGH-400.

Recently, a weak instantiation of GGH was broken in [14], by solving Unique-SVP instances of polynomial gap using LLL up to at least dimension 1000. For many parameters, the numerical gap given in [14] is much lower than what could be hoped from our predictions for LLL, but there is a simple explanation. The problem considered in [14] is actually much easier than a general Unique-SVP problem: it is the embedding of a CVP problem when we already know a nearly-orthogonal basis and the target vector is very close to the lattice. This implies that LLL only performs a size-reduction of the last basis vector, which immediately discloses the solution. This also explains why the LLL running times of [14] were surprisingly low in high dimension.

Recently, a weak instantiation of GGH was broken in [14], by solving Unique-SVP instances of polynomial gap using LLL, up to at least dimension 1000. Surprisingly, for many parameters, the numerical gap of the instances solved in [14] is much lower than what could be hoped from our predictions for LLL. But there is an explanation. The problem considered in [14] is actually much easier than a general Unique-SVP problem: it is the embedding of a CVP problem when we already know a nearly- orthogonal basis and the target vector is very close to the lattice. This implies that LLL is fed with a special input basis (not a random basis), so special that LLL will only perform a size-reduction of the last basis vector, which will immediately disclose the solution. This explains why the LLL running times of [14] were surprisingly low in very high dimension. In other words, the attacks of [14] could even have been carried out without LLL.

5.2 The NTRU lattices

The NTRU cryptosystem [15] is based on the hardness of lattice problems for the so-called NTRU lattices described in [15]. The key generation process of NTRU has changed several times over the past ten years: in the original article [15], the security was based on the hardness of SVP of the NTRU lattices, whereas more recent versions of NTRU are more based on the hardness of CVP in NTRU lattices. To simplify, we compare our predictions with the original description of NTRU based on SVP. In this case, NTRU lattices are essentially characterized by two parameters: N and q such that the dimension is $2N$, the volume is q^N, and there are heuristically N linearly independent shortest vectors of norm a bit smaller than $\sqrt{q}$ (and which are related to the secret key). Such lattices also have $2N$ trivial short vectors of norm q which are already known. Because NTRU lattices do not have a unique shortest vector, it is not clear if this fits any of the models of Section 3. But if we ever find the shortest vector, we will have found a non-zero vector smaller than q, which means solving Hermite-SVP for a suitable factor. Since we know the lattice volume, we can estimate the corresponding Hermite factor for all three historical NTRU parameter sets, as shown in Table 3. On the other hand, Section 3 suggests that we should be able

Value of (N, q)	$(107, 64)$	$(167, 128)$	$(503, 256)$
Hermite factor required	$(1.00976)^{2N}$	$(1.00729)^{2N}$	$(1.00276)^{2N}$

Table 3. Hermite factor required to solve the three historical NTRU parameter sets.

to achieve a Hermite factor of roughly 1.01^{2N}: this means that out of the three NTRU parameter sets, only the first one $(N, q) = (107, 64)$ seems close to what can be achieved in a reasonable time. This parameter set was not supposed to be very secure (see [15]), but to our knowledge, no NTRU-107 lattice has ever been broken by direct lattice reduction. The only successful lattice attack was that of May in 1999 (see [22]), which combined exhaustive search with lattice reduction of smaller lattices. Surprisingly, it was estimated in [15] that NTRU-107 could be broken within a day using raw lattice reduction, but no actual break was reported: the experiments given in [15] only broke slightly smaller values of N. In fact, if we compute the Hermite factor corresponding to each NTRU instance broken in [15] using BKZ, similarly to Table 3, we obtain a Hermite factor of the form c^{2N} where c varies between 1.0116 and 1.0186: such values of c are clearly consistent the results of Section 3.

Still, since $(1.00976)^{2N}$ of Table 3 is very close to the prediction 1.01^{2N}, it seems reasonable to believe that NTRU-107 should be within reach of current algorithms, or small improvements. We therefore made experiments with three NTRU-107 lattices generated at random. Out of these three, only one was broken with BKZ: during the computation of BKZ-25, the shortest vector was found, but BKZ-25 did not even terminate. But BKZ did not succeed with the other lattices, and we stopped the computation after a few days. We then tested a stronger reduction algorithm on all three lattices, inspired by Figure 13:

– We partially reduce the NTRU-107 lattice with BKZ with increasing blocksize for a few hours.
– We project the lattice over the orthogonal complement of the first 107 vectors (we chose 107 based on the GSL slope): this gives a 107-dimensional projected lattice L' whose shortest vectors might be the projections of the initial 214-dimensional lattice L.
– We run BKZ on the projected lattice L' with increasing blocksize until an unusually short vector is found: because L' has much smaller dimension L, Figure 13 implies that we can run much higher blocksize. In practice, we could reach blocksize 40. If the short vector is the projection of one of the shortest vectors of L, we can actually recover a shortest vector of L.

This experiment worked for all three NTRU-107 lattices: we were always able to recover the secret key, using BKZ of blocksize between 35 and 41 on the projected lattice, and the total running time was a few hours. By comparison, raw BKZ reduction only worked for one of the three lattices. This confirms that the Hermite factor prediction 1.01^n gives a good idea of what can be reached in practice. And knowing better the limits and the performances of current algorithms might help to design better ones.

Acknowledgements Part of this work was supported by the Commission of the European Communities through the IST program under contract IST-2002-507932 ECRYPT.

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. of 28th STOC*, pages 99–108. ACM, 1996.
3. M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proc. of 29th STOC*, pages 284–293. ACM, 1997.
4. M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proc. 33rd STOC*, pages 601–610. ACM Press, 2001.
5. D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. In *Proc. of Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 1–11. IACR, Springer, 1999.

6. H. Cohn and N. Elkies. New upper bounds on sphere packings. I. *Ann. of Math. (2)*, 157(2):689–714, 2003.
7. D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. of Cryptology*, 10(4):233–260, 1997. Revised version of two articles from Eurocrypt '96.
8. N. Gama, N. Howgrave-Graham, H. Koy, and P. Q. Nguyen. Rankin's constant and blockwise lattice reduction. In *CRYPTO*, pages 112–130, 2006.
9. N. Gama, N. Howgrave-Graham, and P. Q. Nguyen. Symplectic Lattice Reduction and NTRU. In *Advances in Cryptology – Proceedings of EUROCRYPT '06*, volume 4004 of *Lecture Notes in Computer Science*, pages 233–253. Springer, 2006.
10. N. Gama and P. Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *STOC'08: Proc. of the 40th Annual ACM Symposium on Theory of Computing*. ACM, 2008. To appear.
11. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. Cryptology ePrint Archive, Report 2007/432, 2007. http://eprint.iacr.org/ To appear in STOC '08.
12. O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Proc. of Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. IACR, Springer, 1997.
13. D. Goldstein and A. Mayer. On the equidistribution of Hecke points. *Forum Math.*, 15(2):165–189, 2003.
14. D. Han, M.-H. Kim, and Y. Yeom. Cryptanalysis of the Paeng-Jung-Ha cryptosystem from PKC 2003. In *Public Key Cryptography - Proc. PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 107–117. Springer, 2007.
15. J. Hoffstein, J. Pipher, and J. Silverman. NTRU: a ring based public key cryptosystem. In *Proc. of ANTS III*, volume 1423 of *LNCS*, pages 267–288. Springer-Verlag, 1998. First presented at the rump session of Crypto '96.
16. N. A. Howgrave-Graham and N. P. Smart. Lattice attacks on digital signature schemes. *Des. Codes Cryptogr.*, 23(3):283–290, 2001.
17. R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proc. of 15th STOC*, pages 193–206. ACM, 1983.
18. P. Klein. Finding the closest lattice vector when it's unusually close. In *Proc. of SODA '00*. ACM–SIAM, 2000.
19. J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. *Journal of the Association for Computing Machinery*, January 1985.
20. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Ann.*, 261:513–534, 1982.
21. L. Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity*, volume 50. SIAM Publications, 1986. CBMS-NSF Regional Conference Series in Applied Mathematics.
22. A. May and J. H. Silverman. Dimension reduction methods for convolution modular lattices. In *Proc. of CALC '01*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.
23. D. Micciancio and S. Goldwasser. *Complexity of lattice problems*. The Kluwer International Series in Engineering and Computer Science, 671. Kluwer Academic Publishers, Boston, MA, 2002. A cryptographic perspective.
24. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302 (electronic), 2007.
25. P. Q. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto '97. In *Proc. of Crypto '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 288–304. IACR, Springer, 1999.

26. P. Q. Nguyen and I. E. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *J. Cryptology*, 15(3):151–176, 2002.
27. P. Q. Nguyen and D. Stehlé. Floating-Point LLL Revisited. In *Advances in Cryptology – Proceedings of EUROCRYPT '05*, volume 3494 of *Lecture Notes in Computer Science*, pages 215–233. Springer, 2005.
28. P. Q. Nguyen and D. Stehlé. LLL on the average. In *ANTS*, pages 238–256, 2006.
29. P. Q. Nguyen and J. Stern. The two faces of lattices in cryptology. In *Proc. of CALC '01*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.
30. P. Q. Nguyen and J. Stern. Adapting density attacks to low-weight knapsacks. In *Advances in Cryptology – Proceedings of ASIACRYPT '05*, volume 3788 of *Lecture Notes in Computer Science*, pages 41–58. Springer, 2005.
31. P. Q. Nguyen and T. Vidick. Sieve algorithms for the shortest vector problem are practical. *J. of Mathematical Cryptology*, 2008. To appear.
32. A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In *Proc. of Cryptology and Computational Number Theory*, volume 42 of *Proc. of Symposia in Applied Mathematics*, pages 75–88. American Mathematical Society, 1989.
33. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. Cryptology ePrint Archive, Report 2007/279, 2007. http://eprint.iacr.org/ To appear in STOC '08.
34. O. Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942 (electronic), 2004.
35. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 84–93, New York, 2005. ACM.
36. C.-P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.
37. C.-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Math. Programming*, 66:181–199, 1994.
38. C.-P. Schnorr and H. H. Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Proc. of Eurocrypt '95*, volume 921 of *Lecture Notes in Computer Science*, pages 1–12. IACR, Springer, 1995.
39. V. Shoup. Number Theory C++ Library (NTL) version 5.4.1. Available at `http://www.shoup.net/ntl/`.