NCICIT 2013: 1st National Conference on Intelligent Computing and Information Technology, November 21, CUET, Chittagong-4349, Bangladesh

Analysing Progressive-BKZ Lattice Reduction Algorithm

Md. Mokammel Haque[1], Mohammad Obaidur Rahman[2] and Josef Pieprzyk[1]
[1]Department of Computing, Macquarie University, Australia.,
[2]Department of Computer Science & Engineering, Chittagong Univ. of Engg. & Tech., Bangladesh
Email: malin.mokammel-haque@mq.edu.au; obaidur_91@cuet.ac.bd; josef.pieprzyk@mq.edu.au

***Abstract*— BKZ and its variants are considered as the most efficient lattice reduction algorithms compensating both the quality and runtime. Progressive approach (gradually increasing block size) of this algorithm has been attempted in several works for better performance but actual analysis of this approach has never been reported. In this paper, we plot experimental evidence of its complexity over the direct approach. We see that a considerable time saving can be achieved if we use the output basis of the immediately reduced block as the input basis of the current block (with increased block size) successively. Then, we attempt to find pseudo - collision in SWIFFT hash function and show that a different set of parameters produces a special shape of Gram-Schmidt norms other than the predicted Geometric Series Assumptions (GSA) which the experiment suggests being more efficient.**

Keywords*: *Lattice reduction, BKZ, Gram-Schmidt vectors, SWIFFT.

I. INTRODUCTION

Lattice of our interest is integer or point lattice; discrete subgroup of R^n. It is defined as the integer linear combinations of some vectors (b_1,..., b_d) of the form: $\sum_{i=1}^{d} x_i.b_i$ where xi $\in$ Z. The vectors (b_1,..., b_d) are linearly independent and called a basis of the lattice. The number of vectors d is the lattice dimension.

The goal of lattice reduction is to find good and quality bases that include short enough vectors. SVP or Shortest Vector Problem is the most common and fundamental lattice problem on which many cryptosystems rely on. In lattice-based cryptanalysis one typically look for an approximate variant of SVP, i.e. to obtain a vector $\leq \alpha \times \lambda_1$ (L), where d is the dimension of lattice L, $\alpha \geq 1$ is the approximation factor and λ_1 (L) is the first minimum (the length of the shortest nonzero vector of L). Other important lattice problems are CVP (Closest Vector Problem) and SIVP (Shortest Independent Vector Problem).

Lattice problems are considered as a key element in many areas of computer science as well as cryptography. In particular, public key cryptosystems based on knapsack problem, special setting of RSA and DSA signatures schemes; encryption schemes based on LWE(Learning with error) problems; fully homomorphic cryptosystems are the most important.

Hash function SWIFFT [1], consists of following compression function:

$$\sum_{i=1}^{m} a_i.x_i \in R$$

where $R = Z_q[x]/(x^n+1)$ is the ring of polynomial, $a_i \in R\ (\forall i)$ are m fixed multipliers and $x_i \in R\ (\forall i)$ with coefficients $\in\{0, 1\}$ are the input of length mn. Other parameters of this function include n be a power of 2 and a prime modulus p>0. The algebraic formulation of SWIFFT is equivalent to a lattice of dimension mn and the security of such function depends on the infeasibility of finding relatively short vectors in such a lattice. Finding a collision in this function means to look for a nonzero vector xi $\in$ R with coefficients are in {−1, 0, 1} such that $\sum_{i=1}^{m} a_i.x_i$ =0 mod p. In other words, if we can find a vector by reducing the lattice (considering a sublattice of dimension m'n' $\leq$ mn will be enough) of Euclidean length $\leq \sqrt{mn}$, then it is a *pseudo-collision.*

The most efficient (in terms of reduction quality) BKZ lattice reduction algorithm can able to find such collision in $2^{O(d^2)}$ time complexity. Both theoretically and experimentally we show the variant BKZ-Progressive can do so in reduce amount of time. Precisely, in section 4 we give a heuristic argument that it has enumeration time complexity of nearly $2^{O(d \log d)+O(d)}$ in section 5 we plot experimental results to support this proposition and in section 6 we provide its average runtime complexity for finding pseudo-collision comparing with BKZ. We also showed that it can achieve the quality of reduction as good as BKZ.

II. Related Work

Lattice reduction technique has been studied from the language of quadratic forms (by Hermite in 1850 and then by Korkine-Zolotarev in 1873, together combine HKZ reduction) to analyze current cryptosystems. But it is not seemed to have much improvement since Lenstra-Lenstra-Lovász seminal LLL [2] algorithm has been published in 1982. This polynomial time algorithm is very fast with moderate reduction quality to solve shortest vector problem (SVP). In [3], Schnorr introduced the block concept of HKZ reduction to produce the best reduction algorithm (BKZ) in practice until today. This algorithm gets much slower when block size increases but can achieve approximation ratio (Hermite factor) upto$\approx 1.011^n$ while LLL can achieve roughly upto$\approx 1.022^n$ according to [4]. The practical BKZ algorithm is reported in [5] and has been since widely studied by researchers related to thisfield. Both LLL and BKZ is implemented in NTL package [6] and usually used as a base of any relevant experiments.

Quasi-HKZ reduction technique proposed by Ravi Kannan [7] is another celebrated work that achieve best theoretical bound on enumeration procedure(lattice reduction technique often combines with enumeration routine, for example in BKZ reduction algorithm). An improved analysis of this algorithm is reported in [8] by Stehle´and Hanrot. Recent significant improv ement is done by Gama et al. in [9] where they introduce extreme pruning concept for enumeration. This leads lower success probability to find a short enough vector but gain in overall by saving considerable time. Gama and Nguyen's experiments done in [4] suggest some actual behaviour (output quality v. running time) of lattice reduction algorithms to reduce the gap between theoretical analysis and practical performance of lattice algorithms.

We believe there are two variants of applying BKZ progressively. The first variant uses same lattice basis with increasing block sizes to look for a particular lattice reduction quality. In [10] Buchmann and Lindner used this approach to find sub-lattice collision in SWIFFT, similar approach is used in [11] by Lindner and Peikert to extrapolate BKZ runtime for LWE parameters. The other variant (feed forward approach, i.e. the output of current block is used as the input of next larger block) that used as a preprocessor of the enumeration routine in [12] designated as recursive-BKZ. Our definition of Progressive BKZ is believed to be substantiated the later variant.

III. Preliminaries

A. Gram-Schmidt Orthogonalization

The purpose of lattice reduction is to search for orthonormal basis i.e. look for the unit vectors that are pairwise orthogonal in the basis. Gram-Schmidt orthogonalization method can always transfer a basis **B** = (b_1,..., b_d) into orthogonal basis $\mathbf{B}^*$=(b_1^*,..., b_d^*) as follows:

$$b_1^* = b_i$$
$$b_i^* = b_i \sum_{i>j} \mu_{i,j} b_j^*$$

where $\mu_{i,j} = (b_i, b_j)/(b_j^*, b_j^*)$ for all $i \neq j$. We can also define the orthogonal Gram-Schmidt vectors as $b_i^* := \pi_i(b_i)$ for all non-negative i≤d, where π_i denotes the orthogonal projection over the orthogonal supplement of the linear span of b_1, ..., b_{i-1} [see [13] for details]. Every lattice reduction algorithm uses this process to find the reasonably short lattice vectors.

B. Input and Output basis

The basis that is chosen for SWIFFT can be represented as the following matrix:

$$\mathbf{B} = \begin{pmatrix} I & 0 \\ H & p.I \end{pmatrix}$$

where the m ∗ n dimensional lattice is symbolized as a $m*n \times m*n$ matrix. H is a $n \times n*(m-1)$ skew circulant matrix in Zp = {0… p−1}; p> 0 is the prime for the SWIFFT parameters. I is the $n*(m-1) \times n*(m-1)$ identity matrix. Right bottom is the n × n scalar matrix and right top is a $n*(m-1) \times n$ dimensional zero matrix.

According to [9], sufficiently random reduced bases except special structure have a typical shape for main algorithms like LLL and BKZ and the shape depends on the ratio $q = \|b_{i+1}^*\| / \|b_i^*\|$ of Gram-Schmidt vectors. This follows Schnorr's *Geometric Series Assumption* (GSA)[14]. The running time of the algorithms depends on q. The BKZ-Progressive also produces same shape as BKZ for certain parameter choice $(m' \leq m)$ of the input basis. As m increases (fixing the n') the $\|b_i^*\|'s$ produce a constant 1 consistently, i.e. the ratio q is 1 too. Seemingly, we can say as long as the basis does not produce special structure the shape is typical. From the discussion of [11] we predict the q reaches its lower bound 1. Now on we will mention the first case as the typical behavior of $\left\|b_i^*\right\|$ as well as q and the second case as special behavior. We discuss both cases while analyzing BKZ-Progressive performance and SWIFFT in later sections.

C. *Enumeration:*

Enumeration is a technique that is used alongside lattice reduction for most of the major algorithms. If we imagine a lattice L as a tree then enumeration procedure is simply an exhaustive (depth-first) search for integer combination of the nodes (or vectors) within a upper bound in λ_1(L). This procedure (also implemented in NTL) run in exponential time opposed to the lattice reduction technique itself (which run in polynomial time).

IV. THEORETICAL MOTIVATION FOR BKZ-PROGRESSIVE

Block-Korkine-Zolotarev reduction *a.k.a* BKZ reduction is the most successful lattice reduction algorithm in practice. Schnorr and Euchner [5] introduced the following dentition of BKZ reduction.

A. *BKZ Reduction*

Let β be an integer such that 2≤ β<d. A lattice (L) basis (b_1,..., b_d) is β-reduced if it is size reduced and if (for i =1, ...,d−1)

$$\left\|b_i^*\right\| \leq \lambda_1(\Pi,(L(b_{i,....,}b_{\min(i+\beta-1,d)})))$$

Ravi Kannan provides an algorithm that computes HKZ reduced bases to solve SVP. The main idea of Kannan's algorithm is to spend more time on pre-computing a basis of excellent quality before calling the enumeration procedure. Our BKZ-Progressive approach inspired from Kannan for pre-processing a basis of subsequently increasing block size can be defined as follows.

BKZ-Progressive Reduction: Let BKZ (d, β) denotes the BKZ algorithm running on a lattice dimension d with block size β. Instead of reducing BKZ(d, β) in the first place it will reduce the basis with smaller block size upto β with BKZ(d, 2), BKZ(d, 4), ..., BKZ(d, β).

B. *Heuristic*

A d-dimensional lattice reducing with BKZ-Progressive approach requires enumeration complexity of approximately $2^{O(k \log k)}$, where k is the block size.

Proof. For BKZ reduction with block size k the recent theoretical analysis (see Theorem 2 in [15]) shows that

$$\log q \underset{\sim}{<} \frac{\log(\gamma)}{k-1}$$

where $\gamma \leq c_h * k$ is the Hermite constant in dimension k (for constant c_h). The number of BKZ-k iterations sufficient to get this log q is $\underset{\sim}{<} (c' * n^3 / k^2)$ for constant c' . Since progressive BKZ applies for i =2, 4, ..., k − 2 BKZ(i + 2), on an input basis which is the output of BKZ(i), we expect the enumeration time for BKZ(i + 2) to be (from the heuristic estimation described in [13] for solving SVP using enumeration)

$$_q c(i+2)^2 {}_{=2} \log(c_h.i)/(i-1).c.(i+2)^2 {}_{=2} c \log(c_h.i)(i+O(1))$$

where c is another constant. Since ≤c'* n^3 / (i + 2)^2 iterations needed for BKZ (i+2) for i =2, 1,...,,k −2, the total enumeration time (T) for *BKZ-Progressive* would be

$$T \leq \sum_i c' \frac{n^2}{(i+2)^2} 2^{c \log(c_h, i)(i+O(i))}$$

$$\leq \frac{k}{2} c'.n^3.2^{c \log(c_h, k)(k+O(1))} \underset{-}{\sim} 2^{O(k.\log k)}$$

V. EXPERIMENTAL RESULTS

Experiments are done in 2.67 GHz Corei5 (64 bit) Intel processor machine. The BKZ algorithm and associate enumeration routine is the NTL implementations offloating -point version.

A. *Reduction Quality and Shape of* $\left\|b_i^*\right\|$

The reduction quality of lattice basis depends on the Gram-Schmidt vector $\left\|b_i^*\right\|$, for a good reduction algorithm the value of $\left\|b_i^*\right\|$ does not decay too fast. The parameter $q = \left\|b_{i+1}^*\right\| / \left\|b_i^*\right\|$ is the measure of a lattice reduction quality. According to [13], for LLL reduced basis q ≅ 1:04 (in high dimension) and for BKZ-20 reduction the value is equivalent to 1.025. The slope of the fitted logarithmic linear curve of $\left\|b_i^*\right\|$ is also a measure of reduction quality.

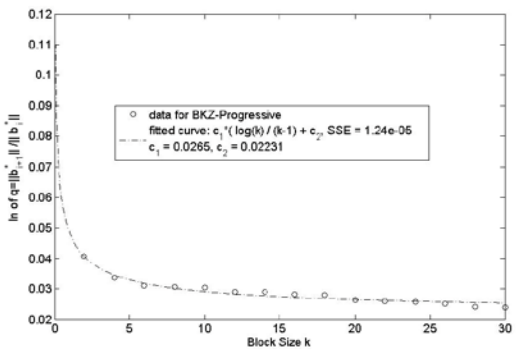


Figure 1. Reduction quality in BKZ-Progressive on an average.

Table 1: Root-*Hermite factor* that can be achieved in BKZ-Progressive reduction with dimension 120.

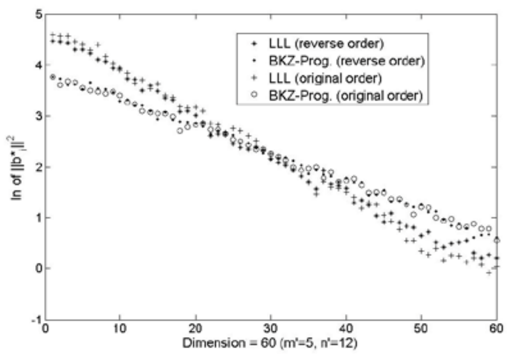
k	*q*	*δ*
2	1.04154	1.02038
4	1.03415	1.01678
20	1.02672	1.01316
22	1.02637	1.01298
24	1.02608	1.01284
26	1.02538	1.01250
28	1.02453	1.01208
30	1.02423	1.01194
32	1.02390	1.01177

In BKZ reduction approach the block size is the parameter that controls the reduction quality. Larger block size gives better reduction in increase of runtime. That means the ratio q should get lower when the block size increases. Experiments done by Gama and Nguyen in [9] (using CJLOSS lattice) found slope = −0.085 for LLL reduction and −0.055 for BKZ-20 reduction (in dimension 110) considering $\|b_i^*\|^2$ instead of $\|b_i^*\|$. Figure 1 show in our case, how the log q (the slope) decreases in increase of block size. A fitted curve for the data points also satisfy that log q $\simeq$ log k/(k−1) (which supports our theoretical assumption for enumeration time).

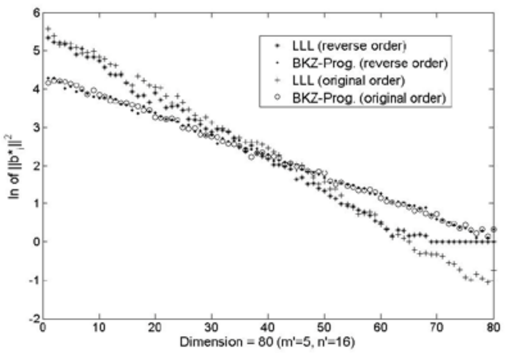
The better reduction directs us to the closer approximation to the shortest vector. The best current *Hermite factor* $(\delta^d = \|b_1\| / \sqrt[d]{vol(L)}$ is reported in [4] is about 1.0109^d for BKZ-28 reduction. As we know

$$q^{\frac{(d+1)}{2}} = \delta^d$$

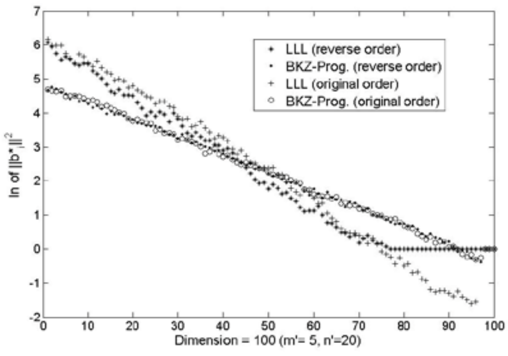
for lattice dimension d, the relation between $q = \|b_{i+1}^*\| / \|b_i^*\|$ and *root-Hermite factor* is approximately $\delta \sim \sqrt{q}$. Table 1 shows our experimental outcomes for BKZ-Progressive reduction. The tabulated values are of q, and *root-Hermite factor* for different increasing block sizes(k) of dimension 120.



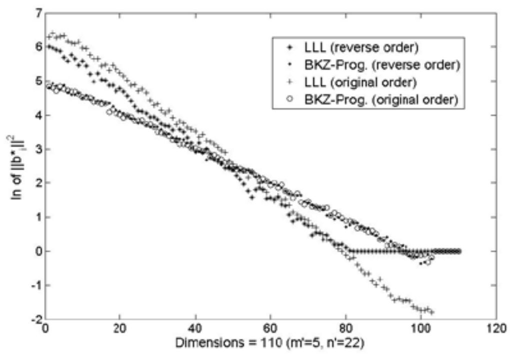
(a)



(b)



(c)



(d)

Figure 2. Shape of $\left\|b_i^*\right\|$ curve changes as dimension increases.

It is to be said that the slope of the fitted $\left\|b_i^*\right\|$ curve will be independent of lattice dimension. But it is true as long as the Gram-Schmidt norms $\left\|b_i^*\right\|$ behaves typically. For our SWIFFT input basis setting we performs experiments to see how the actual shape of $\left\|b_i^*\right\|$ looks like after different reduction approach in increase of dimension. We want to see in what dimension the transition (typical to special case) occur and whether the shape of $\left\|b_i^*\right\|$ changes differently when we consider change in input basis vector's order. Figure 2 plots this idea. Let us say i' is the dimension after which a transition from typical to special case occurs i.e. $\left\|b_i^*\right\|$ (for i>i') becomes constant 1. Also, the current input setting (as in section 3) is considered as original order of basis vectors and a reverse setting of these vectors is considered as reverse order. For original order i' denotes as i' and for reverse order i'_{now}.

For BKZ-Progressive output, the curves of log $\left\|b_i^*\right\|$ v.i are about the same for both orders, and are straight line of gradient log (q) for i $\leq i'_{old}$ $\log\left\|b_i^*\right\| = 0$ for both orders.

For LLL output, in the reverse order, $\log\left\|b_i^*\right\|$ is approximately a straight line with gradient log (q) for $i \leq i'_{new}$, where i'_{new} is the value of line for which the straight line intersects zero. For $i > i'_{new}, \log$ $\left\|b_i^*\right\|$ =0 in this order. In case of original order, $\log\left\|b_i^*\right\|$ is a straight line of gradient log (q) for i in the interval $i'_{new} > i \leq i'_{old}$ (with same values of log (q) and i' as in reverse order). For $i > i'_{old}, \log\left\|b_i^*\right\| = 0$ For $i \leq i'_{new}, \log\left\|b_i^*\right\|$ is roughly a convex curve above the straight line of reverse order, intersecting this straight line at $i = 1\, and = i'_{new} + 1 (i.e. \log\left\|b_1^*\right\|)$ is approximately same for original and reverse orders and $\log\left\|b_i^*\right\|$ =0 for both orders). Also, i' is about the same for BKZ and LLL output (see (c) and (d) of figure 2).

B. *Enumeration Time Analysis*

We know BKZ reduction procedure implemented in NTL consists of two main parts; enumeration segment which performs exhaustive search in enumeration tree and the reduction segment (based calculate the enumeration time per iteration or block. For progressive approach of BKZ-k reduction, while BKZ-2, BKZ-4,...,BKZ-k reduction is performed subsequently, we consider the last block's enumeration time (i.e. BKZ-k) to compare with that of BKZ only approach. Figure 3 gives a comparison graphs for these two approaches. We consider here dimension 120 with a Schnorr-Ho¨rner pruning [16] parameter 1 to allow larger block sizes.

Least squarefitting for the data points of these approaches generate different curvefitting models. In case of BKZ, thefitted m odel is f(k)=0.017 ∗ k^2 − 1.1 ∗ k + 16 with SSE (Sum of Squares due to Error) equivalent to 0.98. On the other hand, BKZ-Progressive fits model f(k)=0.44 ∗ k ∗ ln(k) − 1.8 ∗ k +5.9 with SSE = 0.25. An alternative model f(k)=0.062∗ k ∗ ln(k) − 11.25 also fits quite well with SSE only 0.64.

The above models are generated for special case of $\left\|b_i^*\right\|$, for a typical case scenario we get a similar model for BKZ-Progressive approach (f(k)=0.61∗ k ∗ ln(k) −2.6 ∗ k + 14 with SSE = 0.006). A comparison of enumeration time for both cases is listed on the Table 4 in Appendix A.

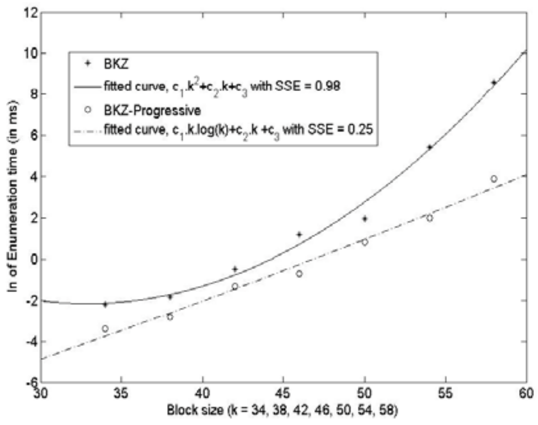


Figure 3. Average enumeration time/block for BKZ and BKZ-Progressive

It shows an improvement over the Peikert-Lindner work in [11] where they used BKZ with block size k directly on LLL reduced basis (default NTL-BKZ implementation), hence resulting in expected BKZ block enumeration time exponential in k^2. We preprocess the basis before applying BKZ-k by progressive BKZ procedure, resulting in a lower on the exhaustive search condition) performs either Trivial, Non Trivial or No Operations reductions (see NTL-BKZ implementation for details). During such a search for the short vector, the enumeration segment is executed number of times. This is the total number of iterations for a BKZ-k reduction. Knowing the total time for those iterations for a block size k, we can expected BKZ block enumeration time exponential in $k * \log(k)$.

VI. CRYPTANALYSIS EXPERIMENTS ON SWIFFT

Again, to obtain *pseudo-collision* we need tofind a vector of norm $\|b_1\|$ (usually thefirst vector in the reduced basis) sssuch that $\|b_1\| \le \sqrt{mn}$. Following [13] we can calculate the value of $\|b_1\|$ for certain parameters (d,n,p and q as their usual meaning) as follows:

$$\|b_1\| \approx q^{(d \quad 1)/2} vol(L)^{1/d} \approx q^{(d \quad 1)/2} \sqrt[d]{p^n} \qquad (1)$$

Where $vol(L)$ is the volume of lattice equivalent to the determinant of it $(= p^n)$. The usual parameters choice for SWIFFT are *m = 16, n = 64 and p = 257.*

A. *Models for Typical and Special Case*

In typical case the graph of log of $\|b_1\|$ *versus i (for i = 1, ..., d)* is a straight line with gradient *log q* (that depends on the block size *k).*This means that

$$\log\|b_i^*\| = \log\|b_1\| - (i-1) * \log q \qquad (2)$$

Since we know that

$$\sum_{i=1}^{d} \log\|b_i^*\| = \log(vol(L)) \qquad (3)$$

Substituting (1) in (2) gives us

$$\log\|b_1\| = (d-1)/2 * \log + 1/d * \log(vol(L)) \, (4)$$

This is the relation we have in equation (1). From (4) and (2), we can see that the quantity *1/d * log(vol(L))* is approximately equal to $\|b_i^*\|$ *for i ~ (d −1)/2, i.e.* the straight line crosses the value *1/d*log(vol(L))* approximately in the middle. This means the lowest value of the line, log $\|b_1\|$ is about *1/d*log(vol(L)) − (d+ 1)/2**log q. So, the condition for the typical case should be that $\log\|b_d^*\| \ge 0$.

Then for the special case, log $\|b_i^*\|$ falls with a gradient

log q for i =1, ..., i′ (where i′ <d) and $\log\|b_i^*\|$ *=0 for* $i \ge i'$. Then using (3) we get the following relation in place of (4)

log $\|b_1\|$ *=(i′ − 1)/2 * log q +1/i′ * log(vol(L))* (5)

Similarly to the above, (5) means that $\log\|b_i^*\|$ *=0= 1/i′ * log(vol(L) − (I′ − 1)/2 * log q,* which is quadratic equation in *i'* as a function of log q and vol(L). By solving this equation we get a model for *i'*. These have been used tofind the *pseudo-collision* in following section.

B. *Pseudo*-Collision *Parameters*

Micciancio and Regev in [17] first observed experimentally that shortest vector of length $l = 2^{2\sqrt{n \log p \log \delta}}$ can be obtained for a optimal dimension $d = \sqrt{n \log p / \log \delta}$. Based on this idea, in our case we can only satisfy the condition of *pseudo-collision* for much smaller n′ (n = 20). The other parameters we consider are δ =1.0117 and p = 257 fixed in our case. For larger n, a successful collision is not possible if we restrict these parameters. A smaller choice of parameter p and δ can obviously find collision for larger n (as well as d). In fact, in [10] a smaller choice of modulus p has been actually considered for *pseudo-collision* (see section 6.2) estimate.

We certainly cannot decrease the value of δ as it is reported from [4] as optimum for current best algorithms and also our experiments (in table 1) support this. However, it

Table 2: Choice of parameters for finding *pseudo-collision* in special case.

D	*m'*	*n'*	*P*	*k*	*q*	$\|b_1\|$	$\|b_1\|'$
96	6	16	257	4	1.03415	11.54	9.70
108	6	18	257	10	1.03081	11.60	10.10
120	6	20	257	12	1.02946	12.52	10.58
140	7	20	257	14	1.02934	12.53	11.57
151	7	22	257	18	1.02811	13.55	12.28
168	7	24	257	22	1.02637	13.89	12.85
175	7	25	257	21	1.02608	11.19	13.22
182	7	26	257	28	1.02538	15.60	12.20

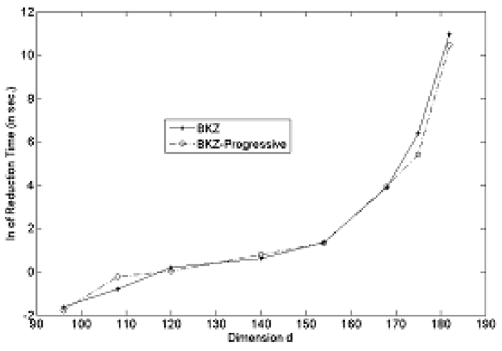


Figure 4. Average runtime for *pseudo*-collision.

would be somewhat different if we considered the special case shape of $\|b_i^*\|$ and extract gradient *log q* from this shape instead of straight line model. But a close investigation reveals that a situation like special case happens when the input vectors are smaller in length (i.e. for larger m and smaller *n'*) in the basis.

Experimental estimate of *pseudo-collision* is listed in table 2 where eight different instances can launch successful attack in feasible time. We plot experimentally derived norm of shortest vector value in column $\|b_1\|$ ', $\|b_1\|$ entries are for theoretically derived shortest vector value. It is calculated by the *i'* <*d* from experimental data and then plugging into the model for special case (section 6.1). We see that the experimental result is better than the theoretical one. We can reach upto *n = 27* within the time around 9.5 hours. We record the runtime to obtain pseudo-collision in both BKZ and BKZ-Progressive approach. Figure 4 shows the comparison graph.

C. *Pseudo*-Collision *in Typical case of* $\|b_i^*\|$

However, it might be interesting to see what experiment can achieve in perfect typical case condition.We found the following entries (in table 3) that satisfy the pseudo-collision constraint. To derive the value of q for corresponding larger

Table 3: Choice of parameters for finding *pseudo-collision* in typical case.

d	*m'*	*n'*	*P*	*k*	*q*
90	5	18	257	26	1.02538
100	1	25	79	28	1.02153
114	3	38	29	44	1.02210
120	3	40	29	50	1.02120

value of *k* we used linear extrapolation. In this case, it is hard to get a *pseudo-collision* as we cannot able to increase the value of $m' \leq m$ much (as its give shape like special case) and if we increase n we rarely find a short vector as volume of lattice jumps high. Again, in this particular situation we need to decrease modulus p to become successful.

The runtime for first two instances (d = 90 and 100) of BKZ reduction requires approximately 185 and 1170 seconds on average. On the other hand, BKZ-Progressive can reduce the instances for 92 and 575 seconds respectively on an average. So, the time improvement is about a factor of two. For other instances, as the block size and dimension both become increasingly high it is unexpected to get collision in feasible time.

References

[1] V. Lyubashevsky, D. Micciancio, C. Peikert and A. Rosen. *SWIFFT: A Modest Proposal for FFT Hashing*. In Fast Software Encryption(FSE) 2008, LNCS, pp. 54-72. Springer-Verlag, 2008.

[2] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lova´sz. *Factoring polynomials with rational coefficient s*. Math. Ann., 261(4): 515-534, 1982.

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

[4] N. Gama and P. Q. Nguyen. *Predicting lattice reduction*. In EUROCRYPT, 2008, LNCS 4965, pp. 31-51. Springer, 2008.

[5] C.-P. Schnorr and M. Euchner. *Lattice basis reduction: improved practical algorithms and solving subset sum problems*. Math. Programming, 66:181-199, 1994.

[6] V. Shoup. NTL: A library for doing number theory. Available at http://www.shoup.net/ntl/.

[7] R. Kannan. *Improved algorithms for integer programming and related lattice problems*. In Proc. of 15th ACM Symp. on Theory of Computing (STOC), pp. 193-206. ACM, 1983.

[8] G. Hanrot, D. Stehle´. *Improved Analysis of Kannan's Shortest Lattice Vector Algorithm*. In CRYPTO 2007, LNCS 4622, pp. 170-186, 2007.

[9] N. Gama, P. Q. Nguyen, O. Regev. *Lattice enumeration using Extreme Pruning*. In EUROCRYPT, 2010, LNCS, Springer-Verlag, 2010.

[10] J. Buchmann and R. Lindner. *Secure Parameters for SWIFFT*. In INDOCRYPT'09, pp. 1-17, 2009. [11] R. Lindner and C. Peikert. *Better key sizes (and attacks) for LWE-based encryption*. In Proc. of the 11th international conference on Topics in cryptology, CT-RSA'11, pp. 319-339, Springer-Verlag, 2011.

[12] Y. Chen and P. Q. Nguyen. *BKZ 2.0: Better Lattice Security Estimates*. In ASIACRYPT'11, LNCS 7073, pp. 1-20.IACR, 2011.

[13] P. Q. Nguyen and V. Brigitte. *The LLL Algorithm: survey and*

application. Chapter-2 (Hermites Constant and Lattice Algorithms). ISBN 978-3-642-02294-4, 2010.

[14] C.-P. Schnorr. *Lattice reduction by random sampling and birthday methods*. In STACS'03, LNCS, vol. 2607, pp. 145,156. Springer, 2003.

[15] G. Hanrot, X. Pujol and D. Stehle´. *Analyzing Blockwise Lattice Algorithms using Dynamical Systems*. Accepted to CRYPTO 2011.

[16] C.-P. Schnorr and H. H. Ho¨rner. *Attacking the Chor-Rivest cryptosystem by improved lattice reduction*. In EURO-CRYPT95, LNCS 921, pp. 1-12,Springer-Verlag, 1995.

[17] D. Micciancio and O. Regev. *Post Quantum Cryptography*. Chapter Lattice based Cryptography. Springer-Verlag, 2009.

APPENDIX A

Table 4: Average enumeration time/block comparison for typical and special case (BKZ-Progressive Reduction, Prune = 1).

	Enumeration Time(ms)	
Block Size(k)	***Typical case***	***Special case***
34	0.086	0.033
38	0.16	0.06
42	0.41	0.27
46	1.36	0.50
50	5.26	2.35
54	23.33	7.62
58	145	48