Improved Pump and Jump BKZ by Sharp Simulator

Leizhang Wang *[1], Wenwen Xia[1], Geng Wang[2], Baocang Wang[1], and Dawu Gu[2]

[1]Xidian University
[2]Shanghai Jiaotong University

Abstract

The General Sieve Kernel (G6K) implemented a variety of lattice reduction algorithms based on sieving algorithms. One of the representative of these lattice reduction algorithms is *Pump* and jump-BKZ (pnj-BKZ) algorithm which is currently considered as the fastest lattice reduction algorithm. The pnj-BKZ is a BKZ-type lattice reduction algorithm which includes the jump strategy, and uses *Pump* as the SVP Oracle. Here, *Pump* which was also proposed in G6K, is an SVP sloving algorithm that combines progressive sieve technology and dimforfree technology. However unlike classical BKZ, there is no simulator for predicting the behavior of the pnj-BKZ algorithm when jump greater than 1, which is helpful to find a better lattice reduction strategy. There are two main differences between pnj-BKZ and the classical BKZ algorithm: one is that after pnj-BKZ performs the SVP Oracle on a certain projected sublattice, it won't calling SVP Oracle for the next nearest projected sublattice. Instead, pnj-BKZ jumps to the corresponding projected sublattice after J ($J \geq 1$) indexs to run the algorithm for solving the SVP. By using this jump technique, the number of times that the SVP algorithm needs to be called for each round of pnj-BKZ will be reduced to about $1/J$ times of original. The second is that pnj-BKZ uses *Pump* as the SVP Oracle on the projected sublattice. Based on the BKZ2.0 simulator, we proposes a pnj-BKZ simulator by using the properties of HKZ reduction basis. Experiments show that our proposed pnj-BKZ simulator can well predicate the behavior of pnj-BKZ with jump greater than 1. Besides, we use this pnj-BKZ simulator to give the optimization strategy for choosing jump which can improve the reducing efficiency of pnj-BKZ. Our optimized pnj-BKZ is 2.9 and 2.6 times faster in solving TU LWE challenge ($n = 75, \alpha = 0.005$) and TU LWE challenge ($n = 60, \alpha = 0.010$) than G6K's default LWE sloving strategy.

Keywords: pnj-BKZ Simulator, Optimal Jump Strategy, TU LWE Challenge, G6K

1 Introduction

Hard problems on lattice: Learning with Errors problem (LWE) and Short Integer Solution (SIS) Problem play central role in cryptography for constructing cryptographic constructions[1], e.g. quantum-safe public-key encryption/key exchange and signatures schemes: Reg09[2], LP11[3], ADPS16[4], BG14a[5]; fully homomorphic encryption: BV11[6], GSW13[7]; and obfuscation of some families of circuits BVWW16[8]. In order to estimate the concrete security strength of these lattice-based cryptographic schemes, generally, the LWE and SIS problem are first reduced to u-SVP on an embedding lattice, and then calculating the computational cost of lattice reduction algorithm solving the u-SVP on the embedding lattice. Furthermore, the simulator of lattice reduction algorithm was constructed to more accurately predict the behavior of lattice reduction algorithm and to construct a more efficient lattice reduction algorithm.

There are many simulators[9, 10, 11] of lattice reduction algorithms like BKZ, BKZ2.0, Progressive-BKZ which can accurately predict how the quality of output basis change during reducing. pnj-BKZ as currently the fastest lattice reduction algorithm, however there is no accurate simulator for pnj-BKZ when $jump > 1$. Since these simulators of classical lattice reduction algorithms can only

*Corresponding auther: leizhangalpha@163.com

predict the value of *jump* equal to 1. By constructing simulator for pnj-BKZ when $jump > 1$, we can find more efficient lattice reducation strategy to solve these approximate-SVP on lattice like (M)LWE and (M)SIS which are the hard problems that almost all lattice based cryptographic schemes' security based on. Therefore, it is of great significance to study how much improvement the jump technology can bring to the lattice reduction algorithm in order to accurately evaluate the concrete security strength of lattice based cryptographic schemes.

In this paper, we proposed pnj-BKZ simulator by using the information of HKZ basis when $jump \geq 1$. The results of experiments show that we can simulate well that how lattice basis changed in each tour of pnj-BKZ when $jump \geq 1$. In addition, we use this pnj-BKZ simulator to give a optimized *jump* choosing strategy. In the end, using optimized *jump* choosing strategy, we can solve TU LWE challenge [1] ($n = 75, \alpha = 0.005$) 2.9 times faster than that of G6K's default LWE sloving strategy[12].

2 Preliminaries

Definition 1 (Lattice). *A lattice L is generated by a basis $\mathbf{B}$ which is a set of linearly independent vectors $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\} \in \mathbb{R}^m$. We will refer to it as $L(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^{n} z_i \mathbf{b}_i, z_i \in \mathbb{Z}\}$. In this paper the length of $\mathbf{v} \in \mathbb{R}^m$ is the Euclidean norm $\|\mathbf{v}\|$.*

Definition 2 (Gram-Schmidt Basis and Projective Sublattice). *For a given lattice basis $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$, we define its Gram-Schmidt orthogonal basis $\mathbf{B}^* = (\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_n^*)$ by $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b}_j^*$ for $1 \leq j < i \leq n$, where $\mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$ are the Gram-Schmidt coefficients (abbreviated as GS-coefficients). In this paper we use l_i to represent the value of $\log(\|\mathbf{b}_i^*\|)$. The lattice determinant is defined as $det(L(\mathbf{B})) := \prod_{i=1}^{n} \|\mathbf{b}_i^*\|$ and it is equal to the volume $vol(L(\mathbf{B}))$ of the fundamental parallelepiped. We denote the orthogonal projection by $\pi_i : \mathbb{R}^m \to span(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^{\perp}$ for $i \in \{1, 2, \ldots, n\}$. In particular, $\pi_\cdot$ is used as the identity map. We denote the local block by the projective sublattice $L_{[i:j]} := L(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_j))$, for $j \in \{i, i+1, \ldots, n\}$.*

Definition 3 (Shortest Vector). *A non-zero vector in a lattice L that has the minimum norm is called the shortest vector. We use $\lambda_1(L)$ to denote the norm of the shortest vector. The notion is also defined for a projective sublattice as $\lambda_1(L_{[i:d]})$. The shortest vector problem (SVP) is the problem of finding a vector of length $\lambda_1(L)$ on a Lattice. For a function $\gamma(n)$ of a lattice dimension n, the standard definition of γ-approximate SVP is the problem of finding a vector shorter than $\gamma(n) \cdot \lambda_1(L)$.*

Heuristic 1 (Gaussian heuristic). *Given a n-dimensional lattice L with determinant $\det(L)$, the Gaussian heuristic predicts that there are about $\mathrm{vol}(C)/\det(L)$ many lattice points in a measurable subset C of $\mathbb{R}^n$ of volume $\mathrm{vol}(C)$.*

In addition, the length of the shortest vector can be approximated by the radius of a sphere whose volume is $\det(L)$, which can be approximated to $\sqrt{\frac{n}{2\pi e}} \det(L)^{1/n}$ by using Stirling's formula. This is usually called the Gaussian heuristic of a lattice, and we denote it by

$$\mathrm{GH}(L) = \det(L)^{1/n} / V_n(1)^{1/n},$$

where $V_n(1)$ is the volume of unit ball of dimension n.

Definition 4 (BKZ reduction). *A basis is BKZ-β reduced with factor δ, if it is LLL-δ reduced and all Gram-Schmidt vectors satisfy $\|\mathbf{b}_i^*\| = \lambda_1(L_{[i,k]})$, where $k = min(i + \beta - 1, d)$ and d is dimension of lattice. As the BKZ simulation algorithm heavily depends on the BKZ reduction algorithm, this algorithm is explained in the next chapter in more detail.*

[1] TU LWE Challenge presents sample instances for testing algorithms that solve the LWE problem. The main goal of this challenge is to help assessing the hardness of the LWE problem in practice. Here is the link to this challenge: https://www.latticechallenge.org/lwe_challenge/challenge.php

Definition 5 (HKZ reduction)**.** *A basis is HKZ reduced, if it is size reduced and all Gram-Schmidt vectors satisfy* $\|\mathbf{b}_i^*\| = \lambda_1\left(L_{[i,d]}\right)$*, where d is dimension of lattice. This particularly means that* $\mathbf{b}_i$ *is a shortest vector of* $L_{[i,d]}$*.*

Next we introduce the Geometric Series Assumption which is now a standard heuristic assumption for predicting the behavior of lattice reduction algorithms. It is supported by extensive experimental results [13] and states that the norm of the Gram-Schmidt vector decreases geometrically after reduction.

Heuristic 2 (Geometric Series Assumption(GSA))**.** *Let L be a d dimension lattice with basis* $\mathbf{b}_1, \ldots, \mathbf{b}_d$*. After execution of BKZ with block-size* β*, the norms of the Gram-Schmidt vectors satisfy*

$$\|\mathbf{b}_i^*\| = \delta_\beta^2 \cdot \left\|\mathbf{b}_{i+1}^*\right\|,$$

for $1 \leq i \leq d-1$*, and where* $\delta_\beta = \left(\frac{\beta}{2\pi e}\left(\pi\beta\right)^{1/\beta}\right)^{1/(2(\beta-1))}$.

In this paper, we use the speed of solving these standard form LWE problem[2] instances in the TU LWE challenge as a measure of the efficiency improvement of our lattice reduction algorithm. Since these standard LWE problem instances in the TU LWE challenge are standard form LWE problem, we will use the AFG14-type lattice basis to reduce the standard form LWE problem instances in the TU LWE challenge into u-SVP on the embedding lattice. Next, we briefly introduce the process of this reduction.

AFG14[14] reduced standard form LWE problem to a u-SVP on a embedding lattice by consturctiong embedding lattice basis from public information. The form of AFG14 type lattice basis is like:

Definition 6 (AFG14 type lattice)**.**

$$\begin{pmatrix} \mathbf{I}_n & \bar{\mathbf{A}} & 0 \\ 0 & q\mathbf{I}_{m-n} & 0 \\ \mathbf{b} & \ldots & 1 \end{pmatrix}$$

where $(\mathbf{A}, \mathbf{b})$ is LWE instances, and $\bar{\mathbf{A}}$ is row echelon form matrix of pubilc key matrix $\mathbf{A}$. We can see that there is an unusually short lattice vector $(\mathbf{e}, 1)$ in this lattice, and the expected length is about $\sigma\sqrt{m}$.

3 Lattice Reduction Algorithm

Blockwise Korkine-Zolotarev (or BKZ) reduction algorithm is a lattice reduction algorithm whose performance between LLL reduction algorithm and HKZ reduction algorithm introduced by Schnorr and Euchner [15]. The main idea of BKZ algorithm is using β' dimension SVP Oracle to find a d dimension lattice vector $\mathbf{v}_i \in L$ s.t. $\|\pi_i(\mathbf{v}_i)\| = \mathrm{GH}\left(L_{[i:i+\beta'-1]}\right)$, then insert it in i-th position of lattice basis, where $\beta' = \min(\beta, d-i)$, β' dimension SVP Oracle can be enumeration or sieving and the dimension of lattice is d. After each inserting the quality of lattice basis will be improved. Besides, after inserting a new short lattice vector, LLL algorithm is called to vanish the linear dependence. The detailed execution process of BKZ can be seen in Algorithm 1.

The BKZ simulation algorithm was developed by Yuanmi Chen and Phong Nguyen [10] and can be used to predicts the Gram-Schmidt norms of a lattice basis after N rounds of BKZ-β reduction. The main idea of Yuanmi Chen and Phong Nguyen's simulator is using GH assumption to predict each the value of l_i' by using the information about the value of $l_1, \ldots, l_{i+\beta'-1}$ and the value of $l_1', \ldots, l_{i-1}'$, where $i \in \{1, 2, \ldots, d\}$ and l_i' means i-th Gram-Schmidt norms of lattice basis vector after reducation processed block $L_{[i:i+\beta'-1]}$. The BKZ simulator can be described as shown in Algorithm 2.

Algorithm 1 The Block Korkin-Zolotarev (BKZ) algorithm with *jump*

Input: A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$, a blocksize $\beta \in \{2, \ldots, d\}$, the value of *jump* (in classical BKZ $jump = 1$), the Gram-Schmidt triangular matrix μ and $\|\mathbf{b}_1^*\|^2, \ldots, \|\mathbf{b}_d^*\|^2$.
Output: The basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is BKZ-β reduced.

$z \leftarrow 0$; $j \leftarrow 0$; LLL $(\mathbf{b}_1, \ldots, \mathbf{b}_d, \mu)$;//LLL-reduce the basis, and update μ
while $z < d - 1$ **do**
  $j \leftarrow j \bmod (d-1) + jump$; $k \leftarrow \min(j + \beta - 1, d)$; $h \leftarrow \min(k+1, d)$; //define the local block
  $\mathbf{v} \leftarrow \text{SVPOracle}\left(\mu_{[j,k]}, \|\mathbf{b}_j^*\|^2, \ldots, \|\mathbf{b}_k^*\|^2\right)$;//find $\mathbf{v} = (v_j, \ldots, v_k) \in \mathbb{Z}^{k-j+1} \setminus \{\mathbf{0}\}$ s.t. $\left\|\pi_j\left(\sum_{i=j}^{k} v_i \mathbf{b}_i\right)\right\| = \lambda_1\left(L_{[j,k]}\right)$
  if $\mathbf{b} \neq (1, 0, \ldots, 0)$ **then**
    $z \leftarrow 0$; LLL $\left(\mathbf{b}_1, \ldots, \sum_{i=j}^{k} v_i \mathbf{b}_i, \mathbf{b}_j, \ldots, \mathbf{b}_h, \mu\right)$ 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

Algorithm 2 Simulation of BKZ reduction

Input: The logarithms of the Gram-Schmidt norms $l_i = \log \|\mathbf{b}_i^*\|$, for $i \in \{1, \ldots, d\}$, the desired blocksize $\beta \in \{45, \ldots, d\}$ and the number of tours N to simulate.
Output: A prediction for the logarithms of the Gram-Schmidt norms $l_i' = \log \|\mathbf{b}_i'^*\|$ after N rounds of BKZ reduction.

for $i = 1$ to 45 **do**
  $r_i \leftarrow$ average $\log \|\mathbf{b}_i^*\|$ of a HKZ reduced random unit-volume 45-dimensional lattice
end for
for $i = 46$ to β **do**
  $c_i \leftarrow \log\left(\mathrm{V}_i(1)^{-1/i}\right) = \log\left(\frac{\Gamma(i/2+1)^{1/i}}{\pi^{1/2}}\right)$
end for
for $j = 1$ to N **do**
  flag $\leftarrow$ true //flag to store whether $L_{[k,d]}$ has changed
  for $k = 1$ to $d - 45$ **do**
    $\beta' \leftarrow min(\beta, d - k + 1)$ //Dimension of local block
    $h \leftarrow min(k + \beta - 1, d)$ //End index of local block
    $\log(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l_i'$
    if *flag* = true **then**
      if $\log(\mathrm{V})/\beta' + c_{\beta'} < l_k$ **then**
        $l_k' \leftarrow \log(\mathrm{V})/\beta' + c_{\beta'}$
        flag $\leftarrow false$
      end if
    else
      $l_k' \leftarrow \log(\mathrm{V})/\beta' + c_{\beta'}$
    end if
  end for
  $\log(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l_i'$
  for $k = d - 44$ to d **do**
    $l_k' \leftarrow \frac{\log(\mathrm{V})}{45} + r_{k+45-d}$
  end for
  for $k = 1$ to d **do**
    $l_k \leftarrow l_k'$
  end for
end for
return $l_{1,\ldots,d}$

Before we introduce pnj-BKZ, we first give a description of the subroutine *Pump* algorithm that pnj-BKZ needs to call frequently. *Pump*[12] is an SVP solving algorithm that combines dimforfree technology [16] and progressive-sieving technology [17]. In G6K *Pump* is usually used as a SVP Oracle to find short lattice vectors whose projection vector is the shortest in corresponding projection sublattice. The detailed description of *Pump* we gived in Algorithm 3.

Algorithm 3 *Pump*

Input: $\mathbf{B}, \kappa, \beta, ds = f, \text{stn} = 30$
Output: **B**

$r := \kappa + \beta; l := max\left\{\kappa + f + 1, r - \text{stn}\right\}; ilb := \kappa; \text{L} := \emptyset;$
$\text{B}_{\pi[\kappa,r]} := \text{LLL}\left(\text{B}_{\pi[\kappa,r]}\right);$
//Phase="init";
$\text{L} := \text{gauss sieve}\left(\text{B}_{\pi[l,r]}, \text{L}\right);$
//Phase="up";
while $l > \kappa + f$ **do**
    $\text{L} := \left\{\text{EL}\left(\mathbf{v}, 1\right) \middle| \mathbf{v} \in \text{L}\right\}, l := l - 1;$
    $\text{L} := \text{sieve}\left(\text{B}_{\pi[l,r]}, \text{L}\right);$
end while
//Phase="down";
while $d > 1$ & $ilb < \kappa + ds$ **do**
    BL := best lifts (L); //score all the vectors in best lifts list of L, and score each $\mathbf{v}_i$ with $score\left(\mathbf{v}_i\right) := \theta^{-i} \frac{\|\mathbf{v}_i\|}{\|\mathbf{b}_i^*\|}$;
    if BL $\neq \emptyset$ **then**
        ii := BL.index (max (BL));//Find the best scoring position;
        Insert $\mathbf{v}_{\text{ii}}$ into the basis $\text{B}_{\pi[\kappa,r]}$;
        $ilb := \text{ii} + 1;$
    else
        $\text{L} := \left\{\text{SL}\left(\mathbf{v}, 1\right) \middle| \mathbf{v} \in \text{L}\right\};$
    end if
    $\text{L} := \text{sieve}\left(\text{B}_{\pi[l,r]}, \text{L}\right);$
    $l := l + 1;$
end while
return B

The pnj-BKZ algorithm is a BKZ-like lattice reduction algorithm that uses *Pump* as the subroutine for finding the shortest vector on the projected sublattice. Unlike the classic BKZ algorithm, pnj-BKZ performs jumps with a jump greater than 1. More specificly, after executing the SVP algorithm on a certain block $B_{[i:i+\beta'-1]}$, instead of calling SVP algorithm on the nearest block $B_{[i+1:i+\beta']}$, the SVP algorithm will be used on the $B_{[i+J:i+\beta'+J-1]}$ block after jumping J indexes. The following is the detialed description of pnj-BKZ.

Algorithm 4 pnj-BKZ

Input: $\mathbf{B}$, β, f_{extra}, $jump = 1$

Output: $\mathbf{B}$

$f := min\left\{max\left\{0, \frac{\beta-40}{2}\right\}, \lfloor 11.5 + 0.075\beta \rfloor\right\} + f_{extra}$;
$ds := f + 3$; $\beta := \beta + f_{extra}$;
$\mathbf{B}$=LLL $(\mathbf{B})$;
for $i \in \left\{1, \ldots, \frac{d+2f-\beta}{jump}\right\}$ **do**
  if $1 \leq i \leq \frac{f+1}{jump}$ **then**
    $\kappa, \beta', f' := 1, \beta - f + jump \cdot i - 1, jump \cdot i - 1$
  else if $\frac{f+1}{jump} \leq i \leq \frac{d-\beta+f}{jump}$ **then**
    $j := jump \cdot i - f$
    $\kappa, \beta', f' := j, \beta, f$
  else
    $j := jump \cdot i - (d - \beta + f)$
    $\kappa,\ \beta',\ f' := d - \beta + j,\ \beta - j + 1,\ f - j + 1$
  end if
  $\mathbf{B}_{\pi[k:\beta'+k-1]} \cdot \mathbf{v}_i = Pump\left(\mathbf{B}_{\pi[k:\beta'+k-1]}, \kappa, \beta', f', ds\right)$
  $\mathbf{B}$=LLL $(\mathbf{B})$
end for
$\mathbf{B}$=$Pump\,(\mathbf{B}, d - \beta + f + 1, \beta, f)$
return B

Different from the classic BKZ, pnj-BKZ will adopt the jump strategy, resulting in $J - 1$ hard to estimate l'_i values between jumps, as shown in the following figure 1:

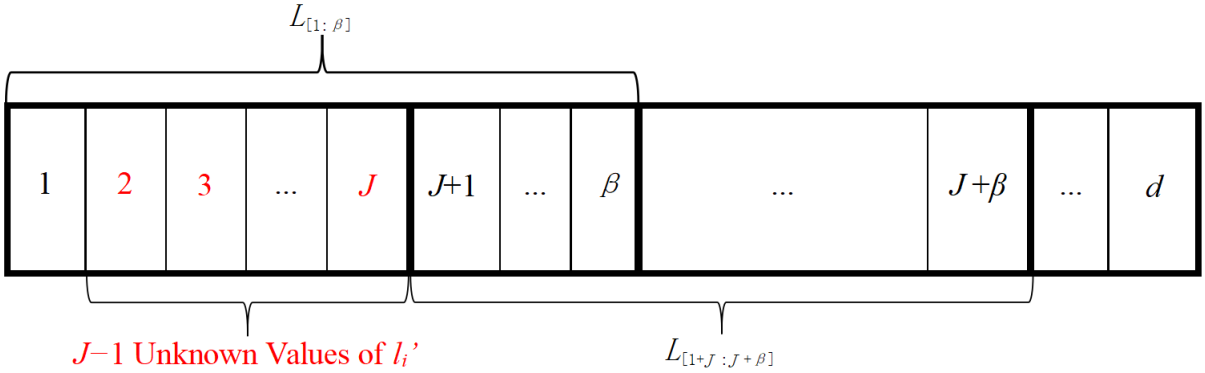


Figure 1: pnj-BKZ with $jump > 1$

Since only knowing the estimation of l'_1, when calculating the l'_{1+J} of the next block, information about values of l'_i is needed, where $i \in \{2, 3, \ldots, J\}$, but the l'_i value will change because of the inserting of l'_1 and it is hard to accurately predict these $J-1$ values of l'_i. Therefore when BKZ-type algorithm with a jump greater than 1, the classic BKZ2.0 simulator cannot be accurately predicted the l'_{1+J} of the next block because lacking necessary information about l'_i , where $i \in \{2, 3, \ldots, J\}$.

4 Simulation of pnj-BKZ

4.1 pnj-BKZ Simulator

The jump strategy which adopted for pnj-BKZ, results in lacking the necessary information about $J - 1$ values of l'_i, $i \in \{2, 3, \ldots, J\}$, when calculating the l'_{1+J} of the next block as shown in Figure

1. We naturally thought of using the information of the HKZ basis to predict these $J-1$ values of l_i' between jumps. More precisely, if each block is HKZ reduced, we can know that

$$l_i' = \log \mathrm{GH}\left(L_{[i:i-(i \bmod J)+\beta-1]}\right) \approx \frac{1}{2}\log\frac{\beta-(i \bmod J)}{2\pi e} + \frac{1}{\beta-(i \bmod J)}\left(\sum_{j=1}^{i-(i \bmod J)+\beta-1} l_j - \sum_{j=1}^{i-1} l_j'\right)$$

where $(i \bmod J) \in \{1, 2, \ldots, J-1\}$ and J is the value of *jump*. In addition, the values from above formula can be calculated by using the Gaussian Heuristic and the information of values of l_k' where $(k \bmod J) \in \{0, 1, \ldots, i-1 \bmod J\}$ which has been obtained. Figure 2 below can intuitively help the readers to understand. Besides, when pnj-BKZ turn on sieving during the pumpdown stage, the HKZ basis is obtained after each *Pump* reduction. Below we give an algorithmic description of the pnj-BKZ simulator utilizing HKZ basis information:

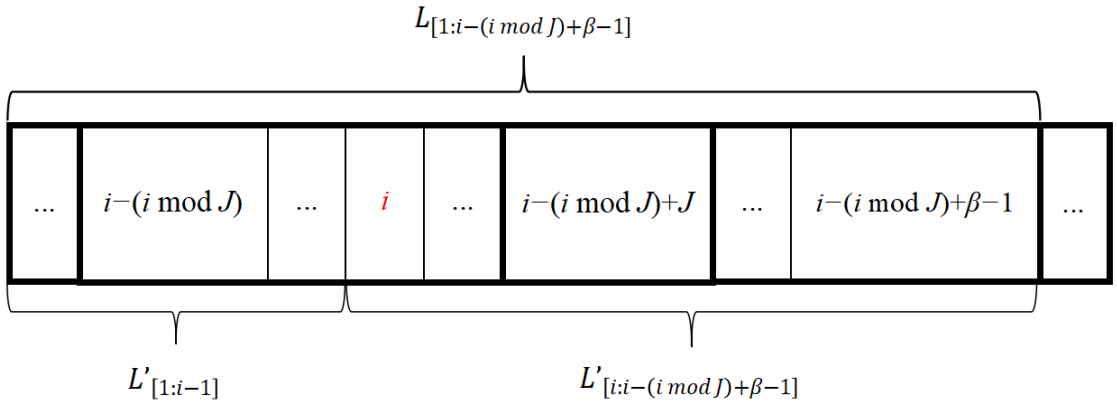


Figure 2: Predict l_i' using the information of HKZ reduction where $(i \bmod J) \in \{1, 2, \ldots, J-1\}$

Algorithm 5 Simulation of pnj-BKZ reduction

Input: The logarithms of the Gram-Schmidt norms $l_i = \log \|b_i^*\|$, for $i \in \{1, \ldots, d\}$, the desired blocksize $\beta \in \{45, \ldots, d\}$, the number of tours N and the size of jump J to simulate.

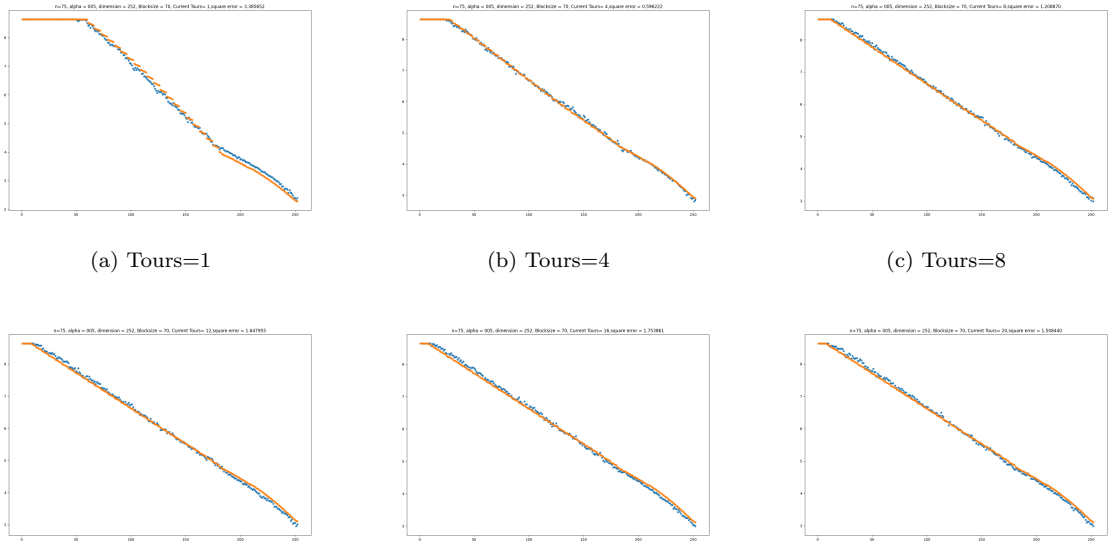
Output: A prediction for the logarithms of the Gram-Schmidt norms $l'_i = \log \|b'^*_i\|$ after N rounds pnj-BKZ reduction with jumpsize is J.

for $i = 1$ to 45 **do**
  $r_i \leftarrow$ average $\log \|b_i^*\|$ of a HKZ reduced random unit-volume 45-dimensional lattice
end for
for $i = 46$ to β **do**
  $c_i \leftarrow \log\left(\mathrm{V}_i\,(1)^{-1/i}\right) = \log\left(\frac{\Gamma(i/2+1)^{1/i}}{\pi^{1/2}}\right)$
end for
for $j = 1$ to N **do**
  $flag \leftarrow$ true //flag to store whether $L_{[k,d]}$ has changed
  for $k = 1$ to $d - \beta$ **do**
    $\beta' \leftarrow min\,(\beta, d - k + 1)$ //Dimension of local block
    if $k \equiv 0\,(mod\ \mathrm{J})$ **then**
      $h \leftarrow min\,(k + \beta - 1, d)$ //End index of local block
      $\log(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l'_i$
      if $flag =$ true **then**
        if $\log(\mathrm{V})/\beta' + c_{\beta'} < l_k$ **then**
          $l'_k \leftarrow \log(\mathrm{V})/\beta' + c_{\beta'}$
          $flag \leftarrow false$
        end if
      else
        $l'_k \leftarrow \log(\mathrm{V})/\beta' + c_{\beta'}$
      end if
    else
      $h \leftarrow min\,(k - (k \bmod \mathrm{J}) + \beta, d)$ //End index of local block
      $\log(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l'_i$
      if $flag =$ true **then**
        if $\log(\mathrm{V})/(\beta' - (k \bmod \mathrm{J})) + c_{\beta' - (k \bmod \mathrm{J})} < l_k$ **then**
          $l'_k \leftarrow \log(\mathrm{V})/(\beta' - (k \bmod \mathrm{J})) + c_{\beta' - (k \bmod \mathrm{J})}$
          $flag \leftarrow false$
        end if
      else
        $l'_k \leftarrow \log(\mathrm{V})/(\beta' - (k \bmod \mathrm{J})) + c_{\beta' - (k \bmod \mathrm{J})}$
      end if
    end if
  end for
  for $k = d - \beta$ to $d - 45$ **do**
    $\beta' \leftarrow d - k$ //Dimension of local block
    $h \leftarrow d$ //End index of local block
    $\log(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l'_i$
    if $flag =$ true **then**
      if $\log(\mathrm{V})/\beta' + c_{\beta'} < l_k$ **then**
        $l'_k \leftarrow \log(\mathrm{V})/\beta' + c_{\beta'}$
        $flag \leftarrow false$
      end if
    else
      $l'_k \leftarrow \log(\mathrm{V})/\beta' + c_{\beta'}$
    end if
  end for
  $\log(\mathrm{V}) \leftarrow \sum_{i=1}^{h} l_i - \sum_{i=1}^{k-1} l'_i$
  for $k = d - 44$ to d **do**
    $l'_k \leftarrow \frac{\log(\mathrm{V})}{45} + r_{k+45-d}$
  end for
  for $k = 1$ to d **do**
    $l_k \leftarrow l'_k$
  end for
end for
return $l_{1,\ldots,d}$

4.2 The actual simulation effect of pnj-BKZ simulator

Executing 20 rounds of pnj-BKZ with different block sizes of 60, 70, 80, 90, and 100 on the TU LWE Challenge instances with jump values of 2, 4, and 6, respectively, and compare the actual reduction results of each round with the simulation results of the pnj-BKZ simulator mentioned above.

Due to space limitations, we present several representative sets of comparison charts between the l_i values predicted by the pnj-BKZ simulator and the actual pnj-BKZ output l_i values. We firstly give these comparison charts about lattice basis: TU LWE challenge $n = 75$, $\alpha = 0.005$; pnj-BKZ parameter: blocksize=70, $jump$=6, pumpdownsieve=True and tours=1,4,8,12,16,20.



(a) Tours=1 (b) Tours=4 (c) Tours=8

(d) Tours=12 (e) Tours=16 (f) Tours=20

Figure 3: β=70, $jump$=6, Pumpdown_sieve=True

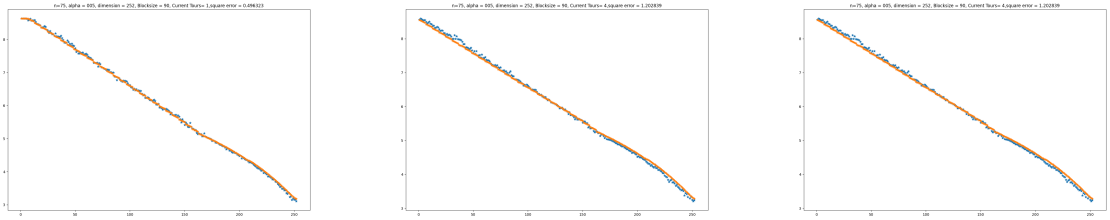
From Figure 3, we can see that as the number of rounds executed by pnj-BKZ gradually increases, these predicted values of l_i given by Algorithm 5 (Orange points in the figure 3) and these actual output l_i values of pnj-BKZ (blue points in the figure 3) gradually approach. We set the sum square error between the predicted value of l_i given by Algorithm 5 and the actual output l_i value of pnj-BKZ as ξ and the smaller value of ξ means the better fitting effect. When Tours=4, the value of ξ is the smallest. The specific value of ξ is shown in the following table 1. Besides the value of ξ is not more than 2 after the number of tours is greater than 8. That is, Algorithm 5 is accurate in predicting the l_i value of pnj-BKZ when $jump$ is greater than 1.

The following comparison charts are lattice basis: TU LWE challenge $n = 75$, $\alpha = 0.005$; pnj-BKZ parameter: blocksize=90,100, $jump$=6, pumpdownsieve=True and tours=1,2,4,8.

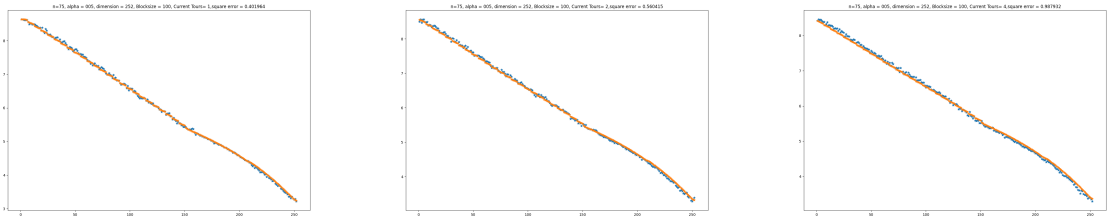
From Figure 4, we can see that when blocksize increased to 90 or 100, our pnj-BKZ simulator proposed in Algorithm 5 can still accurate predict the value of l_i output by pnj-BKZ when $jump$ is greater than 1, where orange points are these predicted values of l_i given by Algorithm 5 and blue points are these actual output l_i values of pnj-BKZ. In addition, the value of ξ is not more than 1 when the number of tours is less than 4. When the number of tours is bigger than 4, the actual reduction effect of pnj-BKZ is not as good as that predicted by the pnj-BKZ simulator in Algorithm 5, but even if the number of tours reaches 20, the value of ξ will not exceed 2. This may be because the Dimforfree technique cannot find the expected short vector when the dimforfree value f is relatively large (the default value f of dimforfree in G6K is larger than the f value given in [Ducas18a][16]). This results in the actual lattice quality not being as good as the quality fitted by the pnj-BKZ simulator.

Table 1: Sum Square Error between Simulation and Actual VSalues

Tours	ξ
1	3.385652
2	2.165373
3	0.915308
4	0.596222
5	0.639893
6	0.664685
7	1.034195
8	1.20887
12	1.647955
16	1.753861
20	1.50844



(a) Blocksizes=90,Tours=1 (b) Blocksizes=90,Tours=4 (c) Blocksizes=90,Tours=8



(d) Blocksizes=100,Tours=1 (e) Blocksizes=100,Tours=2 (f) Blocksizes=100,Tours=4

Figure 4: β=90,100 $jump$=6, Pumpdown_sieve=True

5 Optimizing Jump Strategy

In this section, we use the pnj-BKZ simulator proposed in the previous section and the computational cost model of pnj-BKZ to give the optimal jump value selection strategy.

Before giving the optimal jump selection strategy, we first give the computational cost model of pnj-BKZ. As described in Algorithm 4, pnj-BKZ consists of a series of *Pumps*. Although the initial index κ, blocksize β, and dimforfree values f of these *Pumps* are different, they can be regarded as a $(\beta-f)$-dimensional progressive sieve in general (through the EL operation, a right-to-left progressive sieving is realized). Therefore, ignoring calculating score function, embedding operation and other operations with very small computational cost compared with sieving, the computational cost of pnj-BKZ can be regarded as the sum cost of $\frac{d+2f-\beta}{jump}$ progressive sieving on the $(\beta-f)$-dimension projection sublattice. So we only need to give the computational cost model of the $(\beta-f)$-dimension progressive sieving.

$$\sum_{j=\beta_0}^{\beta-f} 2^{c\cdot j+o(j)} = 2^{c\beta_0}\left(1+2^c+\cdots+2^{c(\beta-f-\beta_0)}\right) \leq 2^{c\beta_0}\cdot\frac{2^{c(\beta-f+1)+o(\beta-f+1)}}{1-2^c} \approx O\left(2^{c(\beta-f)}\right)$$

The β_0 is the dimension of initial sieving in *Pump* (In G6K β_0 usually set as 40). Next, we give the optimal selection strategy for $jump$ by using the pnj-BKZ simulator (Algorithm 5) in section 4.

Algorithm 6 Optimal $jump$ Selection Strategy

Input: $\mathbf{B}^*$, β, $tours$, J_{Bound}
Output: $jump_{optimal}$

$slope_{init} :=$ Slope $(\mathbf{B}^*)$;
for $i \in \{1, 2, \ldots, J_{\text{Bound}}\}$ **do**
  $\mathbf{B}^* =$pnj-BKZ Simulator($basis = \mathbf{B}^*$, $blocksize = \beta$, $Tours = tours$, $jump = i$)
  $slope_i =$ Slope $(\mathbf{B}^*)$
  $\Delta slope_i = slope_i - slope_{init}$
end for
$jump_{optimal} = 0$
$factor = 0$
for $i \in \{1, 2, \ldots, J_{\text{Bound}}\}$ **do**
  if $\Delta slope_i \Big/ \left[\frac{d+2f-\beta}{i}\cdot 2^{c(\beta-f)}\right] > factor$ **then**
    $factor = \Delta slope_i \Big/ \left[\frac{d+2f-\beta}{i}\cdot 2^{c(\beta-f)}\right]$
    $jump_{optimal} = i$
  end if
end for
return $jump_{optimal}$

In algorithm 6, Slope $(\mathbf{B}^*)$ is a function that using the least squares fits the line composed of values of l_i of vectors in $\mathbf{B}^*$ to obtain the slope value which measures the quality of the lattice basis.

To sum up, for pnj-BKZ whose block size is within 60∼100, we give the optimal selection value of $jump$ in an experimental way. For pnj-BKZ whose block size exceeds 100, we use the pnj-bkz simulator and the pnj-BKZ computational cost model to give the optimal selection strategy for $jump$ (Algorithm 6).

6 Solving TU LWE Challenge by using Jump optimized pnj-BKZ

For TU LWE challenge ($n = 75$, $\alpha = 0.005$) and challenge ($n = 60$, $\alpha = 0.010$) [2], we respectively experimented the cost of solving these two challenges by using default strategy of pnj-BKZ which $jump = 1$, and using the Jump optimization selection strategy in Section 4. From table 3, experiments show that the pnj-BKZ with Jump optimization selection strategy can reduce the total cost by 2.6∼2.9 times.

Table 2: Comparison of our optimized pnj-BKZ with the default pnj-BKZ in G6K

	G6K Default pnj-BKZ	Optimized pnj-BKZ
Jump	$jump = 1$	Optimizing $jump$ according to the size of β by Algorithm 6
Sieving During Pumpdown	Turn off	Turn on

[2]More information on this challenge can be found at the link below: https://www.latticechallenge.org/lwe_challenge/challenge.php

Table 3: Efficiency improvement of jump-optimized pnj-BKZ solving TU LWE challenge

	G6K Default pnj-BKZ	Our Optimized pnj-BKZ	Improvement
$(n = 75,\ \alpha = 0.005)$	45109.931	15573.77	2.90
$(n = 60,\ \alpha = 0.010)$	49012.32	18672.443	2.62
The target vector is successfully solved	FALSE	TRUE	NA

All of the experiments in this paper are carried out on the same server. The configuration of our server is as follows: Intel(R) Xeon(R) Silver 4110 CPU, Linux operating system, multi-thread acceleration solution enabled, and the number of threads is uniformly set as 20.

In fact, when G6K's default strategy(CPU version)[3] solves these two TU LWE challenges of $(n = 75,\ \alpha = 0.005)$ and $(n = 60,\ \alpha = 0.010)$, it cannot find the target vector after spending much longer time compared to our optimized pnj-BKZ (it can be see from table3), and even reports an error[4]. If the default LWE solving strategy in G6K can normally execute larger blocks of pnj-BKZ to solve these challenges after debugging, the time cost will only be greater than the time cost in Table 3. In other words, the efficiency gain from our optimized pnj-BKZ in solving these two TU LWE challenges is actually at least greater than 2.8 and 2.6 times.

7 Discussion

In this section we will briefly discuss why Jump and Pumpdown stage turn on sieving will bring good influence on reducing time costing and improving the quality of lattice basis.

Theorem 1. *For the AFG14 lattice L with dimension d and lattice basis quality δ, the lower bound of the dimension of progressive sieving required by Pump to find the target vector $(\mathbf{e}, 1)$ in this lattice L is k. When GSA is established, k is:*

$$k \geq d - \log_{\delta} \frac{(\det |L|)^{\frac{1}{d}}}{\sigma\sqrt{2\pi e}} \tag{1}$$

Proof. From [16] we know that we need the following inequality holds to solve approximate SVP on lattice or projection sublattice : $\lambda_1\left(L_{[d-k+1:d]}\right) \geq \pi_{d-k}\left(\lambda_1\left(L\right)\right)$(The more shorter projection vector are found in projection sublattice the speed of progressive sieving will be quicker [17] and reducing the totall cost of *Pump*). Since under the GH assumption $\lambda_1\left(L_{[d-k+1:d]}\right) = \mathrm{GH}\left(L_{[d-k+1:d]}\right) \approx \sqrt{\frac{k}{2\pi e}}\left(\prod_{i=d-k+1}^{d}\|b_i^*\|\right)^{\frac{1}{k}}$, by using GSA, we have:

$$\begin{aligned}
\lambda_1\left(L_{[d-k+1:d]}\right) &= \sqrt{\frac{k}{2\pi e}}\frac{(\det|L|)^{\frac{1}{k}}\cdot\delta^{\frac{(d-k)(d-k-1)}{k}}}{\|\mathbf{b}_1^*\|^{\frac{d-k}{k}}} \\
&= \sqrt{\frac{k}{2\pi e}}\frac{(\det|L|)^{\frac{1}{k}}\cdot\delta^{\frac{(d-k)(d-k-1)}{k}}}{(\det|L|)^{\frac{d-k}{dk}}\,\delta^{\frac{(d-1)(d-k)}{k}}} \\
\lambda_1\left(L_{[d-k+1:d]}\right) &= \sqrt{\frac{k}{2\pi e}}\frac{(\det|L|)^{\frac{1}{d}}}{\delta^{d-k}}
\end{aligned} \tag{2}$$

Substituting equation 2 into the inequality from [Ducas18a][16], we get:$\sqrt{\frac{k}{d}}\sigma\sqrt{d} \leq \sqrt{\frac{k}{2\pi e}}\frac{(\det|L|)^{\frac{1}{d}}}{\delta^{d-k}}$, where d is the dimension of lattice, σ is the standard deviation of LWE instances and k is the dimension of progressive sieving needed to find the target vector. (In standard form LWE which used in TU LWE challenge, the length of targe vector in embedding lattice is $\sigma\sqrt{d}$).

$$\delta^{d-k} \leq \frac{(\det|L|)^{\frac{1}{d}}}{\sigma\sqrt{2\pi e}}$$

[3]https://github.com/fplll/g6k

[4]python3: hk3_sieve.cpp:1570: std::size_t Siever::hk3_sieve_execute_delayed_insertion(Siever::TS_Transaction_DB_Type&, float&, unsigned int): Assertion 'insertion_start_index > TS_start_queue_original' failed.

$$k \geq d - \log_{\delta} \frac{(\det |L|)^{\frac{1}{d}}}{\sigma\sqrt{2\pi e}}$$

□

From inequality 1 for a certain LWE instances, except for the lattice quality value δ, these right-hand terms of the inequality sign are fixed. Therefore from Theorem 1, it can be clearly seen that when the quality of the lattice is better, the smaller the projected sub-lattice dimension of the sieving needed to find the target vector, that is, the total computational cost of *Pump* is smaller. ·1

Besides, from equation 2 in the *proof* of Theorem 1 we know that the better the quality of lattice basis, the larger the $\lambda_1\left(L_{[d-k+1:d]}\right)$ value will be. Meanwhile, the smaller the projected sublattice dimension k of progressive sieving required to find target vector during *Pump* which leads to smaller computational cost of *Pump*. When we consider *Pump* on a projected sublattice, we can replace d with d' and k with k' in the equation 2. For a projected sublattice, the above analysis is also established. That is, the better the lattice quality of current block where *Pump* working on, the total computational cost of the *Pump* is smaller. Because *Pump* needs to perform progressive sieving (expand from right to left) to find the shortest vector on the current projected sublattice, and it can be seen from equation 2 that the dimension k' of final sieving will decrease with the decrease of δ (This can be understood as, in order to ensure that the value of $\lambda_1\left(L_{[d'-k'+1:d']}\right)$ is slightly larger than the value of $\pi_{d-k}\left(\lambda_1\left(L\right)\right)$, when δ decreases, the dimension k' of progressive sieving needed can also be reduced accordingly).

This explains, to some extent, that when $jump$ is greater than 1, and pnj-BKZ turn on sieving during pumpdown stage, the algorithm can obtain the reduced basis with better quality and faster speed than that of the default pnj-BKZ which $jump = 1$ and not turn on sieving during pumpdown stage. From one hand, $jump$ technique is used which can decrease the number of calls of SVP Oracle to reduce overall overhead and to some extent this also will leads to lattice quality decreased when the value of $jump$ is large. On the anther hand, sieving is turned on during the pumpdown stage to obtain a stronger reduced lattice basis (In fact every block is a d dimension HKZ basis) and make it sensible to jump some index since current block has already been reduced. Therefore if $jump$ is greater than 1, and pnj-BKZ turn on sieving during pumpdown stage, the algorithm can obtain a more reduced lattice basis and according to inequality 2 and we know that this will reducing the total cost of progressive sieving and *Pump*.

8 Conclusion and Future work

In summary, we propose a pnj-BKZ simulator with $jump \geq 1$ based on the properties of the HKZ reduction basis, and the experimental results show that our simulator can well predict the behavior of pnj-BKZ with $jump \geq 1$. Secondly, based on the pnj-BKZ simulator, we optimized pnj-BKZ by optimizing the value of $jump$. The optimized pnj-BKZ can solve the TU LWE challenge by 2.6∼2.9 times faster than the default pnj-BKZ of G6K which $jump = 1$ and not turn on sieving during pumpdown stage.

We will further optimize the optimal selection strategy of $jump$ by more experiments more with different block sizes and the optimal value of $jump$ based on our pnj-BKZ simulator. Then we will try to use this pnj-BKZ simulator to give a LWE solving strategy with stronger reduction effect and less computational cost than the default reduction strategy of G6K. Besides, our simulator is based on the information about HKZ basis, so we will try to construct a simulator of pnj-BKZ without turning on sieving during pumpdown.

References

[1] M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer, "Revisiting the expected cost of solving usvp and applications to lwe," in *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, Eds. Cham: Springer International Publishing, 2017, pp. 297–322.

[2] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *In STOC*. ACM Press, 2005, pp. 84–93.

[3] R. Lindner and C. Peikert, "Better key sizes (and attacks) for lwe-based encryption," in *Topics in Cryptology – CT-RSA 2011*, A. Kiayias, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 319–339.

[4] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key Exchange—A new hope," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 327–343. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim

[5] S. Bai and S. D. Galbraith, "An improved compression technique for signatures based on learning with errors," in *Topics in Cryptology – CT-RSA 2014*, J. Benaloh, Ed. Cham: Springer International Publishing, 2014, pp. 28–47.

[6] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) lwe," in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, 2011, pp. 97–106.

[7] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology – CRYPTO 2013*, R. Canetti and J. A. Garay, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 75–92.

[8] Z. Brakerski, V. Vaikuntanathan, H. Wee, and D. Wichs, "Obfuscating conjunctions under entropic ring lwe." New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2840728.2840764

[9] S. Bai, D. Stehlé, and W. Wen, "Measuring, simulating and exploiting the head concavity phenomenon in bkz," in *Advances in Cryptology – ASIACRYPT 2018*, T. Peyrin and S. Galbraith, Eds. Cham: Springer International Publishing, 2018, pp. 369–404.

[10] Y. Chen and P. Q. Nguyen, "Bkz 2.0: Better lattice security estimates," in *Advances in Cryptology – ASIACRYPT 2011*, D. H. Lee and X. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–20.

[11] Y. Aono, Y. Wang, T. Hayashi, and T. Takagi, "Improved progressive bkz algorithms and their precise cost estimation by sharp simulator," in *Advances in Cryptology – EUROCRYPT 2016*, M. Fischlin and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 789–819.

[12] M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens, "The general sieve kernel and new records in lattice reduction," in *Advances in Cryptology – EUROCRYPT 2019*, Y. Ishai and V. Rijmen, Eds. Cham: Springer International Publishing, 2019, pp. 717–746.

[13] M. Ajtai, "Generating random lattices according to the invariant distribution," *Draft of March*, vol. 2006, 2006.

[14] M. R. Albrecht, R. Fitzpatrick, and F. Göpfert, "On the efficacy of solving lwe by reduction to unique-svp," in *Information Security and Cryptology – ICISC 2013*, H.-S. Lee and D.-G. Han, Eds. Cham: Springer International Publishing, 2014, pp. 293–310.

[15] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," in *Fundamentals of Computation Theory*, L. Budach, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 68–85.

[16] L. Ducas, "Shortest vector from lattice sieving: A few dimensions for free," in *Advances in Cryptology – EUROCRYPT 2018*, J. B. Nielsen and V. Rijmen, Eds. Cham: Springer International Publishing, 2018, pp. 125–145.

[17] T. Laarhoven and A. Mariano, "Progressive lattice sieving," in *Post-Quantum Cryptography*, T. Lange and R. Steinwandt, Eds. Cham: Springer International Publishing, 2018, pp. 292–311.