An Experimental Study of the BDD Approach for the Search LWE Problem

Rui Xu[1(✉)], Sze Ling Yeo[2], Kazuhide Fukushima[1], Tsuyoshi Takagi[3], Hwajung Seo[2], Shinsaku Kiyomoto[1], and Matt Henricksen[4]

[1] KDDi Research Inc., Fujimino, Japan
ru-xu@kddi-research.jp
[2] Institute for Infocomm Research (I2R), Singapore, Singapore
[3] Institute of Mathematics for Industry (IMI), Kyushu University, Fukuoka, Japan
[4] Huawei Technologies, Singapore, Singapore

Abstract. The proved hardness of the Learning With Errors (LWE) problem, assuming the worst case intractability of classic lattice problems, has made it a standard building block in the recent design of lattice based cryptosystems. Nonetheless, a thorough understanding of the security of these schemes from the perspective of existing attacks remains an open problem. In this manuscript, we report our implementation of the Bounded Distance Decoding (BDD) approach for solving the search LWE problem. We implement a parallel version of the pruned enumeration method of the BDD strategy proposed by Liu and Nguyen.

In our implementation we use the embarrassingly parallel design so that the power of multi-cores can be fully utilized. We let each thread take a randomized basis and perform independent enumerations to find the solution instead of parallelizing the enumeration algorithm itself. Other optimizations include fine-tuning the BKZ block size, the enumeration bound and the pruning coefficients and the optimal dimension of the LWE problem. Experiments are done using the TU Darmstadt LWE challenge. Finally we compare our implementation with a recent parallel BDD implementation by Kirshanova et al. [18] and show that our implementation is more efficient.

Keywords: Learning With Errors · Lattice based cryptography · Security evaluation · Bounded Distance Decoding

1 Introduction

Decades of development in the area of lattice-based cryptography have identified two important primitive hard problems, namely, the Shortest Integer Solution (SIS) problem [1] and the Learning With Errors (LWE) problem [24], to be standard building blocks of modern lattice-based cryptosystems.

In this work, we focus on the LWE problem proposed by Regev [24]. LWE has attracted more and more attention since its proposal. Initially LWE problem was reduced to the GAPSVP (the decision version of the shortest vector problem) or

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 253–272, 2017.
DOI: 10.1007/978-3-319-61204-1_13

SIVP (Shortest Independent Vector Problem) under the quantum setting. This means that LWE is considered hard if there are no algorithms to efficiently solve the GAPSVP or SIVP using a quantum computer. Subsequently, the hardness reduction as been sharpened to accept a classic reduction to these standard lattice problems [8]. As such, LWE-based schemes are widely studied as potential primitives in the post-quantum era.

LWE. Let n be a positive integer, denoting the dimension of the lattice related with the LWE problem, q an odd prime, and let $\mathcal{D}$ be an error distribution over the integer ring modulo q, $\mathbb{Z}_q$. Denote by $\mathbf{s}$ a fixed secret vector in $\mathbb{Z}_q^n$ (in this manuscript we adopt the row vector convention to be consistent with software implementation) selected according to the uniform distribution on its support. Let $\mathcal{L}_{n,q,\mathcal{D}}$ be the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ generated by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing an error e according to $\mathcal{D}$ and returning

$$(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$$

in $\mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors in $\mathbb{Z}_q^n$. The search LWE problem is to find the secret vector $\mathbf{s}$ given a fixed number of m samples from $\mathcal{L}_{n,q,\mathcal{D}}$.

Although the intractability of LWE is well established by the reduction proofs, its concrete hardness is far from clear. In this work we follow the approach of Liu and Nguyen [21] to evaluate the performance of the BDD approach for solving LWE problem.

1.1 Our Contribution

In this manuscript, our main contributions include:

- We implement a parallel version of the BDD approach for solving the LWE problem. The implementation features an embarrassingly parallel design where each thread takes a randomized basis and performs an independent enumeration. The advantage of this design is that the power of multi-cores can be fully utilized.
- We give heuristic analysis on how to choose the optimal sub-dimension of the LWE instance. We use the Gaussian heuristic to estimate the cost of a pruned enumeration tree to find better sub-dimension which can reduce the time to solve an LWE instance.
- We compare our implementation with that of Kirshanova et al. [18] and show the advantages of our implementation. Specifically we show that the performance of our parallelization strategy is not limited by Amdahl's Law and the extreme pruning in our implementation brings huge speedup compared with the linear pruning used in the implementation of [18].
- We demonstrate that our implementation solves a couple of instances from the TU Darmstadt LWE challenge.

2 Preliminaries

2.1 Discrete Gaussian Distribution

We first describe the error distribution $\mathcal{D}$ in the LWE problem. In the general situation, any error distribution with small variance is fine for the LWE problem to be hard. However, in this work, similar to many other previous works regarding LWE, we focus on the discrete Gaussian distribution over the ring $\mathbb{Z}_q$ as the error distribution. Let $x \in \mathbb{Z}$. The discrete Gaussian distribution over $\mathbb{Z}$ with mean 0 and width parameter σ, denoted by $D_{\mathbb{Z},\sigma}$ assigns to each $x \in \mathbb{Z}$ the probability proportional to $\exp(-x^2/2\sigma^2)$. The error distribution we consider for the LWE problem is the discrete Gaussian distribution over $\mathbb{Z}_q$, denoted by $D_{\mathbb{Z}_q,\sigma}$, by accumulating the values of the probability mass function over all integers in each residue class mod q. In the original proposal of Regev, the width parameter associated with the moduli q is $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$, where α is the relative error rate. With a slight abuse of notation, we also denote the discrete Gaussian distribution as $\mathcal{D}_{\mathbb{Z}_q,\alpha q}$. When the error distribution of an LWE instance is $\mathcal{D}_{\mathbb{Z}_q,\alpha q}$, we express the LWE instance as $\mathcal{L}_{n,q,\alpha}$.

2.2 Lattice

A lattice in $\mathbb{R}^m$ is a discrete additive subgroup generated by a (non-unique) basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_m)^T$. Equivalently, the lattice $\Lambda(\mathbf{B})$ generated by $\mathbf{B}$ is given by $\Lambda(\mathbf{B}) = \{x | x = \sum_{i=1}^{m} z_i \mathbf{b}_i\}$, where z_i's are integers. Note that by our convention, the vector $\mathbf{b}_i$ in the basis matrix $\mathbf{B}$ is its row vector. The rank of the lattice $\Lambda(\mathbf{B})$ is defined as the rank of the basis matrix $\mathbf{B}$. If the rank of $\Lambda(\mathbf{B})$ equals m, we say that the lattice is full rank. A fundamental notion that lies in various lattice problems is the successive minimal $\lambda_k(\Lambda)$ which is defined to be the smallest real number r such that the lattice contains k linearly independent nonzero vectors of Euclidean length at most r. Specifically, $\lambda_1(\Lambda)$ is the length of the shortest nonzero vector of the lattice Λ.

The lattices we are interested in are a special type of lattices called q-ary lattices which are lattices satisfying $q\mathbb{Z}^m \subset \Lambda \subset \mathbb{Z}^m$. Fix positive integers $n \leq m \leq q$, where n serves as the main security parameter, and q is an odd prime. For any matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$, define the following two lattices.

$$\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x}\mathbf{A} = 0 \mod q\},$$
$$\Lambda_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x} = \mathbf{A}\mathbf{s} \mod q \text{ for some } \mathbf{s} \in \mathbb{Z}_q^n\}.$$

It is easy to check that both $\Lambda_q^{\perp}(\mathbf{A})$ and $\Lambda_q(\mathbf{A})$ are q-ary lattices [23].

2.3 Lattice Reduction

As we have noticed in Sect. 2.2 that a lattice can be generated from different bases, the property of the basis plays a central role in the difficulty of various hard lattice problems. Informally, the more orthogonal the basis is, the easier

the corresponding lattice problems are. As such, many attempts to solve hard lattice problems try to alter (often called reduce in the literature) the given basis in order to get basis which generates the same lattice while at the same time achieves the highest orthogonality possible. We adopt the convention that the first vector $\mathbf{b}_1$ in a reduced basis has the smallest length among the (reduced) basis vectors. After the lattice reduction algorithm, we can use the vector $\mathbf{b}_1$ as an approximation of the shortest vector. Since the determinate of a lattice is invariant under lattice reduction, when the basis is reduced, the length of each basis vector decreases. The common measurement of the quality of a lattice basis is called Hermite factor δ^m defined as: $||\mathbf{b}_1|| = \delta^m \mathtt{vol}(\Lambda)^{1/m}$. We also refer to δ as the root-Hermite factor. A smaller root-Hermite factor typically implies a reduced basis with higher quality.

Lattice reduction algorithms can be viewed as a hierarchy of BKZ [26] based on the parameter blocksize β. The case when $\beta = 2$ is called LLL reduction, which was invented by Lenstra et al. [20]. LLL reduction is proven to run in polynomial time in the lattice dimension and outputs a short vector which is within an exponential factor of the minimal length of a lattice Λ, i.e., $\lambda_1(\Lambda)$. When $\beta = m$, i.e., the full size of the basis, the output basis is HKZ reduced [17] which implies solving the SVP. The situation when k lies in between 2 and m is known as the BKZ-β reduction which is the most referenced reduction algorithm in practice. Chen and Nguyen observed that the running time of BKZ reduction is mainly dominated by the root-Hermite factor δ and is less affected by the dimension m. See Chen and Nguyen [11] for a detailed analysis and their improvements over the standard BKZ as a collection of optimization known as BKZ 2.0. See also Albrecht et al. [2] for a thorough comparison of different estimations of the complexity of BKZ.

2.4 Pruned Enumeration

Gram-Schmidt Orthogonalization. Given a lattice basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_m)^T$, the Gram-Schmidt Orthogonalization of $\mathbf{B}$ is denoted as $\mathbf{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_m^*)$, where $\mathbf{b}_i^*$ is computed as $\mathbf{b}_i^* = \mathbf{b}_i - \sigma_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$ for $i = 1, \ldots, m$, with $\mu_{i,j} = < \mathbf{b}_i, \mathbf{b}_j^* > / ||\mathbf{b}_j^*||^2$ for all $1 \leq j \leq i \leq m$. Denote by $\pi_i(\cdot)$ the orthogonal projection onto $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1})^{\perp}$. Then $\mathbf{b}_i^* = \pi_i(\mathbf{b}_i)$. Also $\pi_i(\mathbf{\Lambda}(\mathbf{B}))$ is an $(m + 1 - i)$-dimensional lattice generated by the basis $(\pi_i(b_1), \pi_i(b_2), \ldots, \pi_i(b_{i-1}))^T$.

Lattice Enumeration. Given a target vector $\mathbf{t}$, a lattice basis $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_m)^T$ and a radius R, lattice enumeration algorithm enumerates over all lattice vectors $\mathbf{v} \in \mathcal{L}$ such that $||\mathbf{v} - \mathbf{t}|| \leq R$ and finds the closest one. The enumeration algorithm enumerates over a search tree leveled by the enumeration depth $k \in [m]$. The root of the search is levelled using $k = 0$ and it represents the target vector. And $k = m$ corresponds to the leaves. The nodes at level k of the search tree consist of all vectors $\mathbf{v} \in \mathbf{\Lambda}(\mathbf{B})$ such that $||\pi_{m+1-k}(\mathbf{t} - \mathbf{v})|| \leq R$. Gama et al. [14] use Gaussian heuristic to approximate the number of nodes in the k-th level of the enumeration tree as

$$H_k = \frac{V_k(R)}{\prod_{i=m+1-k}^{n} ||\mathbf{b}_i^*||^2}, \tag{1}$$

where $V_k(R)$ denotes the volume of a k-dimensional ball of radius R. Then the total number of nodes in the enumeration tree is $N = \sum_{k=1}^{m} H_k$.

Gamma et al. also suggest to use extreme pruning to accelerate the enumeration algorithm. The idea of extreme pruning is by deliberately setting the probability that the solution vector is in the tree after pruning to be very small to cut lots of branches in the enumeration tree. Though the success probability of finding the desired solution becomes quite low, it is compensated by the huge reduction of the enumeration time. Their experiments show an exponential speed up over full enumeration. Formally, pruned enumeration bounds the enumeration tree by limiting the k-level nodes to those vectors $\mathbf{v} \in \mathbf{\Lambda}(\mathbf{B})$ such that $||\pi_{m+1-k}(\mathbf{v} - \mathbf{t})|| \leq R_k$ with R_k denoting the pruned radius and $R_1 \leq R_2 \leq \ldots \leq R_m = R$.

3 Related Work

We consider the search version of the LWE problem in this work. There are mainly three ways to solve the search LWE problem.

1. BKW approach: Blum, Kalai and Wasserman proposed BKW algorithm for the LPN (learning with parity noise) problem. Since LWE can be viewed as a generalization of LPN problem, BKW was also adapted to solve LWE by Albrecht et al. [3].
2. Algebraic approach: Arora-Ge [6] proposed to set up a system of algebraic equations over integers to describe the LWE problem and solve the search problem by solving the equation system. Later, this method was improved by using Gröbner basis techniques [4].
3. BDD approach: This approach views the search LWE problem as a decoding problem in a lattice. We will explain this idea in more details in the following.

Bounded Distance Decoding (**BDD**): Given m samples $(\mathbf{a_i}, c_i)$ following the given LWE distribution $\mathcal{L}_{n,q,\mathcal{D}}$, we organize the input into a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ whose rows constitute the m samples of the vector $\mathbf{a}_i$, and a vector $\mathbf{c} \in \mathbb{Z}_q^m$ whose i-th element is c_i from the i-th sample. Note that $\mathbf{c} = \mathbf{As} + \mathbf{e}$, where $\mathbf{e}$ is the error vector which follows the distribution $\mathcal{D}^n$. When the error distribution of the LWE problem is the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}_q,\alpha q}$, we observe that the length of $\mathbf{e}$ is relatively small since each of its entries is distributed according to the discrete Gaussian. Consider the q-ary lattice

$$\Lambda_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x} = \mathbf{As} \mod q \text{ for some } \mathbf{s} \in \mathbb{Z}_q^n\},$$

induced by $\mathbf{A}$. Then the vector $\mathbf{c}$ is bounded in distance from a vector $\mathbf{v} \in \Lambda_q(\mathbf{A})$. Finding the vector $\mathbf{v}$ from the q-ary lattice is called the **BDD** problem.

One approach to solve the BDD problem is to reduce the BDD problem to unique SVP problem by the embedding technique of Kannan [17] and solve the corresponding SVP problem. Another more common approach is to adapt the well-established Babai's algorithm to solve the BDD problem directly. In this

regard, Lindner and Peikert [22] proposed a variant of Babai's nearest plane algorithm to solve the BDD problem. Bischof et al. [10] and Kirshanova et al. [18] have implemented parallel versions of this algorithm and investigated its practical performance. Liu and Nguyen [21] observed that Lindner and Peickert nearest plane algorithm can be viewed as a form of pruned enumeration where in the former, the pruning strategy bounds the coefficients instead of the usual way of bounding the projection lengths. Further, they propose to use the lattice enumeration with GNR extreme pruning strategy to accelerate the speed of finding the closest vector. This will be the approach we use in our experimental study. We refer the readers to the excellent survey by Albrecht et al. [2] for a comprehensive exploration of the concrete hardness of LWE.

4 Our Implementation

We choose to implement the Liu and Nguyen algorithm to study its practical performance. This algorithm uses enumeration with extreme pruning to solve the BDD problem. Given M samples $\{(\mathbf{a}_i, c_i)\}_{i=1,2,\ldots,M}$ from the LWE distribution $\mathcal{L}_{n,q,\alpha}$, we use the matrix representation to express the LWE problem as $\mathbf{As}+\mathbf{e} = \mathbf{c}$. We outline the algorithm steps as follows:

Algorithm 1. LWE solver using BDD approach

Require: Inputs are $n, M, \alpha, q, \mathbf{A} \in \mathcal{Z}_q^{M\times n}, \mathbf{c} \in \mathcal{Z}_q^M$. The inputs satisfy $\mathbf{As} + \mathbf{e} = \mathbf{c}$ mod q for some hidden secret vector $\mathbf{s} \in \mathcal{Z}_q^n$ and error vector $\mathbf{e} \in \mathcal{Z}^M$ with its coefficients chosen independently according to the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}_q,\alpha q}$.

Ensure: Output the hidden secret vector $\mathbf{s} \in \mathcal{Z}_q^n$ with high probability.

1: Decide an appropriate sub-dimension $\underline{m}$ for the given LWE instance.
2: Choose m random rows from the matrix $\mathbf{A}$ as $\mathbf{A}' \in \mathcal{Z}_q^{m\times n}$ and the corresponding m elements from the vector $\mathbf{c}' \in \mathcal{Z}_q^m$.
3: Compute the basis of the q-ary lattice $\Lambda_q(\mathbf{A}')$ generated by $\mathbf{A}'$, denote it by $\mathbf{B}$.
4: Choose an appropriate block size $\underline{\beta}$ and use BKZ-β to reduce the lattice basis $\mathbf{B}$.
5: Choose an appropriate enumeration radius $\underline{R}$.
6: Compute a set of pruning coefficients and use pruned enumeration to find the closest vector $\mathbf{v}$ to the target vector $\mathbf{c}'$ using the enumeration radius R.
7: If the closest vector $\mathbf{v}$ can be found, then compute the secret vector $\mathbf{s}$ by solving the equation $\mathbf{A}'\mathbf{s} = \mathbf{c}' - \mathbf{v}$ mod q and return $\mathbf{s}$. Otherwise, goto Step 2.

The red and underlined parameters sub-dimension $\underline{m}$, BKZ block size $\underline{\beta}$ and enumeration radius $\underline{R}$ in Algorithm 1 need to be optimized to achieve better performance for our LWE solver. However, it is easy to see that these parameters affect the running time and success probability of our LWE solver in an entangled way. Thus it is a multi-object optimization problem. In general, it is not trivial to solve such a multi-object optimization problem.

We give detailed explanation of the choice we make regarding each step of the algorithm in subsequent subsections.

4.1 Compute the Basis

This is easy linear algebra. Given a matrix $\mathbf{A}$ of size $m \times n$, we want to find the matrix $\mathbf{B}$ which is the basis of the q-ary lattice $\Lambda_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x} = \mathbf{As} \bmod q \text{ for some } \mathbf{s} \in \mathbb{Z}_q^n\}$. Recall that we use the convention of row vectors, so a lattice vector generated by the basis $\mathbf{B}$ can be represented by $\mathbf{zB}$ where $\mathbf{z}$ is a row vector. If we forget for a moment that we are working on the modular ring $\mathbb{Z}_q$, then the basis for $\Lambda_q(\mathbf{A})$ is simply $\mathbf{A}^T$, the transpose of $\mathbf{A}$ since all lattice vectors except for those in $q\mathbb{Z}^m$ can be expressed by an integer linear combination of rows in matrix $\mathbf{A}^T$. To include $q\mathbb{Z}^m$ in order to make it q-ary lattice, we further compute the Hermite normal form of $\begin{bmatrix} \mathbf{A^T} \\ q\mathbf{I}_m \end{bmatrix}$ to get the basis of the q-ary lattice $\Lambda_q(\mathbf{A})$. In other words, $\mathbf{B} = \text{HNF}(\begin{bmatrix} \mathbf{A^T} \\ q\mathbf{I}_m \end{bmatrix})$, where $\text{HNF}(\mathbf{A})$ denotes the row Hermite normal form of a matrix $\mathbf{A}$ removing all zero rows. To see this, first note that $q\mathbb{Z}^m$ itself can be viewed as a lattice with the (m dimensional) identity matrix scaled by q as its basis. Thus, we get $\Lambda_q(\mathbf{A}) = \Lambda(\mathbf{A^T}) \cup \Lambda(q\mathbf{I}_m)$. The last step of our computation relies on the following fact:

Fact 1. Given two lattices $\Lambda(\mathbf{B}_1)$ and $\Lambda(\mathbf{B}_2)$ of the same dimension, the basis for the lattice generated by the union of $\Lambda(\mathbf{B}_1)$ and $\Lambda(\mathbf{B}_2)$ is $\text{HNF}(\begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix})$.

4.2 Enumeration Radius and Basis Randomization

The enumeration radius only affects the running time and success probability of the enumeration part. Consider an LWE instance $\mathcal{L}_{n,q,\alpha}$. Fix m samples from it to get the equation $\mathbf{A}'\mathbf{s} + \mathbf{e}' = \mathbf{c}' \bmod q$. The exact BDD radius is the length of the error vector $\mathbf{e}' \in \mathcal{Z}^m$. Though we do not know the exact value of $||\mathbf{e}'||^2$, we know that its coefficients are generated from the discrete Gaussian distribution $\mathcal{D}_{\mathcal{Z}_q,\alpha q}$. According to the acceptance criteria of the LWE challenge, the requirement is that $||\mathbf{e}'|| \leq 2\sqrt{m}\alpha q$ for an LWE instance $\mathcal{L}_{n,q,\alpha}$ with m samples. Thus one option is to take $R = 2\sqrt{m}\alpha q$ as the **BDD** bound. More generally, we set squared **BDD** bound R^2 as $c \cdot m\alpha^2 q^2$ for some fixed constant c. To approximate the **BDD** bound R, we sampled error vectors according to the discrete Gaussian distribution and record the squared length of the error vector $\mathbf{e}'$. See Fig. 1 for our experiment results. The figure shows the histograms of the distribution of the squared norm of error vector $\mathbf{e}'$ with the bins set as the multiplier between $||\mathbf{e}'||^2$ and $m\alpha^2 q^2$ (i.e., the scalar c). From Fig. 1 we can see that $c = 1.3$ is an appropriate choice for making sure that the closest vector can be found with overwhelming probability and the distribution of the scalar c does not depend on the parameters n, α and m. However, if one chooses $c = 1$ so as to reduce the running time of the enumeration algorithm, then with probability about half, the closest vector can not be found within this radius.

Typical applications of pruned enumeration will first randomize the lattice basis by multiplying the basis matrix with a random unimodular matrix and then apply pruned enumeration to find the desired shortest vector or closest vector. If we adopt this method in our LWE solver, two problems arise.

1. The randomized basis usually has larger entries than the initial basis, thereby adding some burden to the lattice reduction algorithm.
2. If we want to choose a smaller enumeration radius such as setting the scalar c less than 1.3, we might miss the opportunity of finding the closest vector. This is primarily due to the fact that we are working with different bases of a fixed lattice and hence, the error norm is fixed.

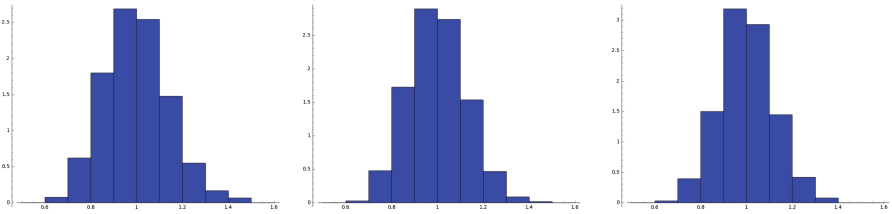
Our randomization is natural and effective. We do not bother to do randomization over a fixed lattice but instead we choose a different lattice each time. In most instances, the number of samples M is larger than the LWE dimension n. It follows that after deciding on a sub-dimension m with $n < m < M$, we can randomly choose m samples from the total M samples to form a different lattice each time. This simple trick solves the two problems discussed above. First the entries of the generated basis are all less than or equal than q. Second, since we randomize over the different m-combinations of the samples, the error vector $\mathbf{e}'$ changes every time. Then we can choose a lower enumeration bound R and be confident that a fixed portion of the trials contribute to error vectors within the bound. For example, according to Fig. 1, if we choose $R^2 = m\alpha^2 q^2$ then the closest vector could be found within this bound with probability about 50 %.

Denote by p_{enum} the success probability of an enumeration algorithm given that the length of the error vector $||\mathbf{e}'||$ is indeed within the enumeration bound R. For simplicity we first consider $||\mathbf{e}'|| \leq 1.3 \cdot m\alpha^2 q^2$. Let $T_{enum}(c)$ be the time of the enumeration algorithm when setting the enumeration radius to be $c \cdot m\alpha^2 q^2$. We can estimate the total time of enumeration to find the closest vector (when setting $c = 1.3$) as $T(1.3) = T_{enum}(1.3)/p_{enum}$. Further assuming that changing the enumeration radius does not affect p_{enum}, we can approximate the success probability of the enumeration algorithm using different enumeration scalars c. For example if we choose $c = 1$ the probability that the error vector is within $1 \cdot m\alpha^2 q^2$ is about 0.5 so we get the total enumeration time of solving the BDD problem as $T(1) = 2 * T_{enum}(1)/p_{enum}$. In particular, choosing $c = 1$ may lead to a faster algorithm if $T_{enum}(1) < T_{enum}(1.3)/2$. Thus, by analyzing the impact of the enumeration radius on the running time of enumeration algorithm, we can choose nearly optimal enumeration radius. Finally, we can use the Gaussian heuristic Eq. (1) to approximate $T_{enum}(c)$ (see below).

4.3 Choose Sub-dimension

In the typical setting of LWE problem, the number of total samples M is bounded by a polynomial of the LWE dimension n. When treating LWE as a lattice problem, an important decision concerns a suitable choice for the dimension of the lattice. The dimension of the lattice equals the number of samples we choose. How many of the total M samples do we use to form the generating matrix $\mathbf{A}'$?

First, we show that if the sub-dimension were chosen too small, the sub LWE problem may not have a unique solution. Consider the following equation $\mathbf{A}'\mathbf{s} + \mathbf{e}' = \mathbf{c}'$. For any choice of $\mathbf{s}$, we can find an error vector $\mathbf{e}'$ satisfying the above equation. However, the LWE problem restricts the length of $\mathbf{e}'$. More



(a) $n = 40$, $\alpha = 0.005$, $m = 100$. (b) $n = 50$, $\alpha = 0.010$, $m = 120$. (c) $n = 60$, $\alpha = 0.015$, $m = 140$.

Fig. 1. Histograms of square length of $\mathbf{e}'$ for different parameters.

precisely, each element of $\mathbf{e}'$ is chosen from the discrete Gaussian distribution with small variance and thus, it can not be too large. TU Darmstat University has held an LWE challenge website[1] similar to the famous SVP challenge. According to Buchmann et al. [9], the acceptance criteria for the correct answer of the LWE problem $\mathcal{L}_{n,q,\alpha}$ with M samples is that $||\mathbf{e}|| \leq 2\sqrt{M}\alpha q$. Based on this criteria, when we choose the sub-dimension to be m, we would also expect to find a secret vector $\mathbf{s}$ such that it leads to a error vector of length less than $2\sqrt{m}\alpha q$. Following the argument of Buchmann et al. [9], we calculate the probability that the sub LWE problem has more than one solution. For a chosen matrix $\mathbf{A}'$ of size $m \times n$, let $\Lambda_q(\mathbf{A}')$ denote the q-ary lattice generated by $\mathbf{A}'$. Recall that $\lambda_1(\Lambda_q(\mathbf{A}'))$ is the norm of the shortest non zero vector in $\Lambda_q(\mathbf{A})$. Assume that we have two solutions for the secret vector $\mathbf{s}_1$ and $\mathbf{s}_2$ satisfying the criteria $\mathbf{A}'\mathbf{s}_1 + \mathbf{e}'_1 = \mathbf{c}' = \mathbf{A}'\mathbf{s}_2 + \mathbf{e}'_2$ and $\mathbf{e}'_i \leq 2\sqrt{m}\alpha q$. Then by the triangle inequality, we have $||\mathbf{A}'(\mathbf{s}_1 - \mathbf{s}_2)|| \leq 4\sqrt{m}\alpha q$. Since $\mathbf{A}'(\mathbf{s}_1 - \mathbf{s}_2)$ is actually a vector in the q-ary lattice $\Lambda_q(\mathbf{A}')$, the fact that the sub LWE problem has more than one solution implies that $\lambda_1(\Lambda_q(\mathbf{A}')) \leq 4\sqrt{m}\alpha q$. On the other hand, Gaussian heuristic tells us that the expected length of the shortest vector of $\Lambda_q(\mathbf{A}')$ is $q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}$. In view of this, in our implementation we choose the sub-dimension m such that the corresponding Gaussian heuristic $q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}$ is larger than $4\sqrt{m}\alpha q$ so that the expected number of solutions is small.

The next question concerns how large the lattice sub-dimension m should be. Note that a large dimension invariably increases the time for the basis reduction. In [15], the authors experimentally showed that for a random input lattice, the root Hermite factor δ after a BKZ-beta reduction is independent of the lattice dimension. The following table shows the root Hermite factor obtained from various m random samples taken from the LWE challenge with $\alpha = 0.005$ under a BKZ-20 reduction averaged over 20 experiments for each pair of m and n (Table 1).

One sees that for each n, the value of δ is approximately 0.128 for the first values of m but deviates from this value for larger values of m. A closer examination reveals that in the latter case, the shortest vectors produced by the

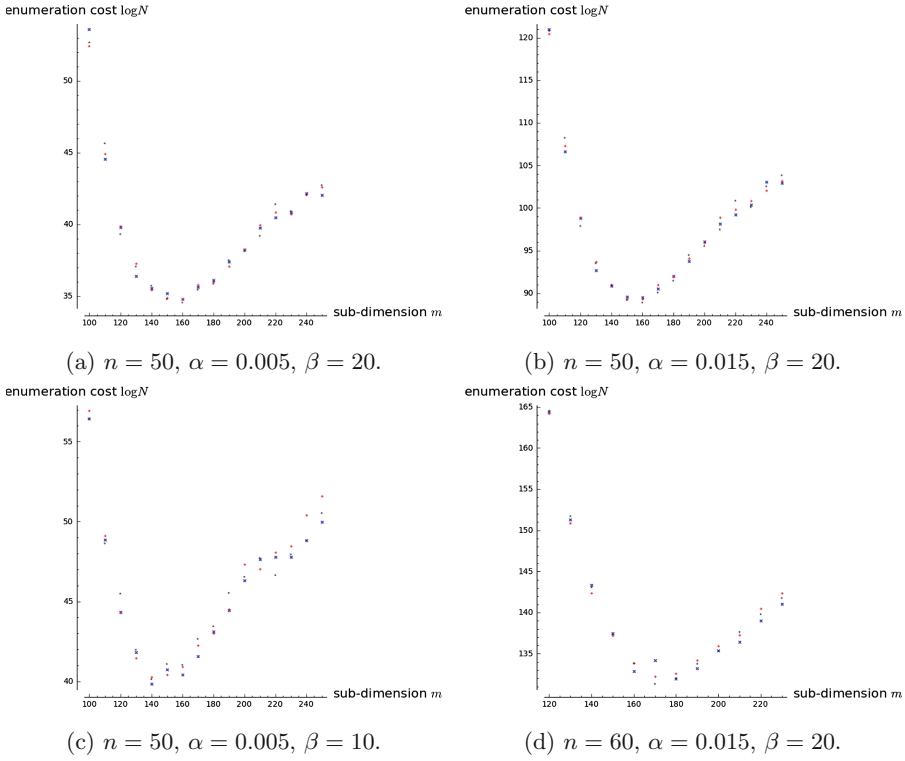
[1] https://www.latticechallenge.org/lwe_challenge/challenge.php.

Table 1. Average root hermite factor for LWE instances with sub-dimension m

n	m										
	100	110	120	130	140	150	160	170	180	190	200
40	1.01272	1.01312	1.01291	1.01289	1.01308	1.01296	1.01160	1.01026	1.00915	1.00821	1.00741
45	1.01290	1.01276	1.01291	1.01301	1.01290	1.01298	1.01309	1.01193	1.01063	1.00954	1.00860
50	1.01282	1.01290	1.01282	1.01291	1.01301	1.01289	1.01309	1.01298	1.01215	1.01090	1.00983
55	1.01285	1.01298	1.01291	1.01291	1.01296	1.01310	1.01296	1.01296	1.01311	1.01229	1.01109
60	1.01281	1.01279	1.01287	1.01291	1.01296	1.01304	1.01301	1.01305	1.01309	1.01312	1.01236
65	1.01286	1.01280	1.01300	1.01304	1.01297	1.01303	1.01298	1.01312	1.01304	1.01322	1.01300

reduction algorithm are the unit vectors scaled by q. In general, we will like to have the lattice reduction to produce vectors with length less than q which tends to suggest that the input vectors are more well-mixed by the reduction algorithm to produce short vectors. In view of this, for a given BKZ-beta reduction, we will select the sub-dimension m such that the predicted shortest vector has length less than q, namely, $\delta^m q^{1-n/m} \leq q$ or, $m \leq \sqrt{n \log q / \log \delta}$, where δ is the expected root Hermite factor. For $\delta = 1.0128$, one checks that this gives the pairs (n, m) to be $(40, 152)$, $(45, 164)$, $(50, 175)$, $(55, 186)$, $(60, 196)$ and $(65, 206)$.

Apart from the lattice reduction, the size of m also affects the enumeration cost. Here, we examine the impact of m for the full enumeration tree. We propose to use the Gaussian heuristic to estimate the enumeration cost, i.e., Eq. (1) to estimate the (full) enumeration cost and to decide the optimal sub-dimension. The total cost is $N = \sum_{k=1}^{m} H_k$. We however do not know how to systematically solve the equation to find an optimal m which minimizes the total cost N. Instead we use numerical calculation to determine the optimal sub-dimension m for fixed BKZ block size β. We plot the total cost $N = \sum_{k=1}^{m} H_k$ for an LWE instance by varying the sub-dimension m. Figure 2 shows the estimated (logarithm of) full enumeration cost for different parameters. We deploy a conversion that for an LWE instance $\mathcal{L}_{n,q,\alpha}$, q is set to be the next prime of n^2 which follows the parameter setting of the LWE Challenge. Comparing Fig. 2a and b, we see that for fixed n and BKZ block size β, the optimal sub-dimension m does not depend on the relative error rate α. Figure 2a and c show the impact of BKZ block size β on the optimal sub-dimension. As we can see, by increasing β, the optimal sub-dimension m also increases and the full enumeration cost decreases for larger β. However, the larger β requires more BKZ reduction time. There is still a need for a trade-off between the BKZ reduction time and enumeration time by setting an appropriate block size β. We discuss this in the next subsection. Finally combining Fig. 2b and d, we can further get the impression that for fixed relative error rate α and BKZ block size β, the larger the dimension of the LWE instance n is, the larger sub-dimension we need to get an optimal performance. One problem of the numerical method to decide on the optimal sub-dimension is that we did not consider the BKZ reduction part. In practice we need to consider the running time of BKZ reduction algorithm, so the actual optimal sub-dimension is usually less than that viewed from the plot. However, the plot can still act as a rough guide to find the optimal sub-dimension. Due to page



(a) $n = 50$, $\alpha = 0.005$, $\beta = 20$.

(b) $n = 50$, $\alpha = 0.015$, $\beta = 20$.

(c) $n = 50$, $\alpha = 0.005$, $\beta = 10$.

(d) $n = 60$, $\alpha = 0.015$, $\beta = 20$.

Fig. 2. Semi-log graph for full cost N. The parameters have the following reference: n is the LWE dimension, α is the relative error rate of the LWE instance and β is the block size for BKZ reduction algorithm used. Different colors stand for different trials. (Color figure online)

constraints, we defer the details of the estimation of enumeration cost and the relation between cost of full enumeration and that of enumeration with extreme pruning in Appendix A.

4.4 Balancing Reduction and Enumeration

Since we use enumeration to solve the BDD problem, we want to first reduce the lattice basis before applying enumeration. BKZ is now the de-facto standard of lattice reduction algorithm in cryptanalysis. We use the BKZ implementation in FPLLL [5] library to perform BKZ reduction.

The quality of the reduced basis and the running time of BKZ reduction algorithm highly depend on the block size β. The choice of an appropriate block size β affects the total running time of our LWE solver. Generally speaking, a larger block size β leads to longer running time of BKZ reduction algorithm but the highly reduced basis will decrease the running time of enumeration. So the folklore is that the optimal block size β should balance the running time

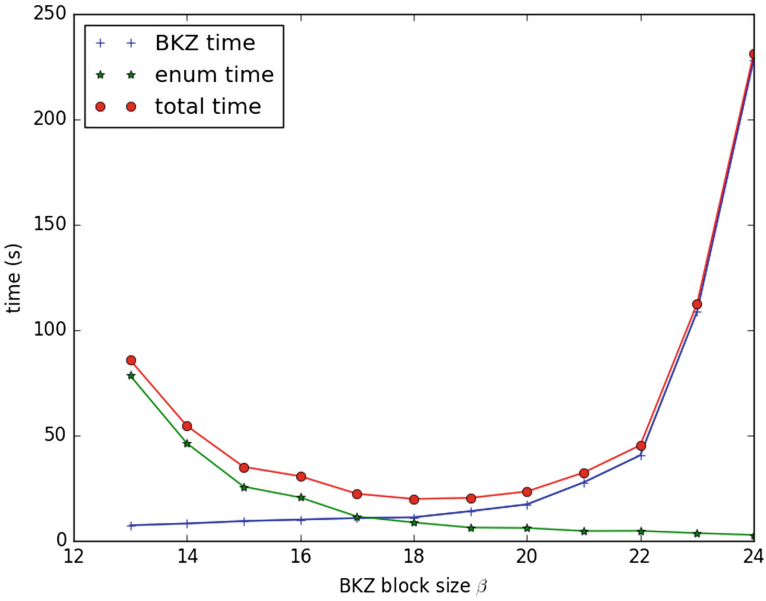


Fig. 3. Running time for BKZ reduction and pruned enumeration.

of BKZ reduction and enumeration. In other words, when the BKZ reduction time and the enumeration time are close to each other, the total running time is minimized. See Fig. 3 for an example. We plot the actual running time of BKZ reduction algorithm and pruned enumeration time for the LWE instance $\mathcal{L}_{40,1601,0.015}$ using sub-dimension $m = 120$. The enumeration radius is set to be $R^2 = 0.8m\alpha^2 q^2$. The figure confirms the folklore that the optimal block size β should roughly balance the running time of BKZ reduction and enumeration. However, finding such optimal block size is not easy, especially for extreme pruning. In our experiments we manually tune the block size β by measuring the running time of the BKZ reduction part and pruned enumeration part.

4.5 Parallelization

Parallelism is ubiquitous in today's program design. We have multi-core CPUs even in our laptops. It is natural to implement the LWE solver algorithm in parallel. One option is to use parallel implementation of enumeration algorithm and parallel implementation of lattice reduction algorithm. Alternatively, One can use a sequential implementation of lattice reduction algorithm and enumeration algorithm but instead launch several threads to solve the BDD problem with different randomized basis. We choose the latter approach for its simplicity and its embarrassing parallelism. Although there are parallel implementations of lattice enumeration algorithms [12,13,16,19], we do not know any public available parallel implementation of BKZ reduction algorithm. Thus if we want to

use parallel implementation of enumeration we might have to use a sequential implementation of BKZ reduction. Amdahl's Law sets a bound on the potential program speedup defined by the fraction of code (p) that can be parallelized as $speedup = \frac{1}{1-p}$. In using the combination of BKZ reduction and enumeration to solve the SVP or CVP problem, it is common knowledge that when the running time of BKZ reduction part and the enumeration part are roughly equal, the total running time is minimized (refer to the previous section and Fig. 3). If we want optimal performance then the fraction of parallelizable code would be about 1/2. It follows that regardless of how many threads are used, the speedup can be at most 2. We can circumvent this by using a small block size for the BKZ reduction, or plugging in the parallel enumeration into the BKZ reduction, but those methods are either complicated or do not achieve optimal performance gain.

In our implementation we use the embarrassingly parallel design to let each thread work on a different randomized basis, and thus there is no load balance issue. In order to achieve best performance we carefully choose the BKZ block size so that the BKZ reduction time is comparable to the enumeration time.

5 Experimental Results

Our implementation is written in C++, using the library FPLLL for BKZ reduction and lattice enumeration. Our program is compiled using gcc 5.4.0 on a desktop running Ubuntu 14.04 LTS. We test our LWE solver using the instances from the LWE challenge website. We use extreme pruning [14] for lattice enumeration as suggested by Liu and Nguyen [21]. Gamma et al. [14] suggest using numerical approximation to generate optimal pruning coefficients by fixing the successful probability and seeking for minimum overhead. Aono [7] also describes how to compute the optimal pruning coefficients. We follow Aono's approach to compute the optimal pruning coefficients in our implementation. We are preparing to release the source code after further optimization. At this moment, it is available upon request.

5.1 LWE Challenge

TU Darmstadt held a LWE challenge project. The challenge provides LWE instances with different parameters. The LWE challenge instance is identified by two parameters: the LWE dimension n and the relative error rate α. The other parameters of an LWE instance are set as follows:

- Moduli q is set as the next prime of n^2;
- Number of samples is set as $M = n^2$;
- Error distribution is set as the discrete Gaussian distribution with width parameter $\sigma = \alpha q$, i.e., the distribution $\mathcal{D}_{\mathbb{Z}_q,\sigma}$.

Using our implementation described in the preceding section, we solved several instances from the LWE Challenge website. Please refer to Table 2 for the

Table 2. Results on solving some instances from the LWE Challenge website

LWE parameters			BKZ reduction		Enumeration		#Trials	Total time
n	α	m	β	t	c	t		
40	0.005	100	10	-	1.3	-	2	4 s
40	0.01	120	10	4 s	1.3	4 s	2	16 s
40	0.015	120	18	12 s	0.8	10 s	819	18403 s
40	0.02	140	32	10.9 d	1.3	27.1 d	-	38 d
45	0.005	120	5	3 s	1.3	1 s	6	23 s
50	0.005	120	15	5 s	1.0	2 s	5	35 s
60	0.005	140	28	27 h	0.8	24 h	-	51 h

Note that the instances $\mathcal{L}_{40,0.02}$ and $\mathcal{L}_{60,0.005}$ take quite long time thus we use parallelized version of the solving algorithm, and we do not record the number of trials for these two instances.

detailed recording of the LWE parameters, the block size we used for BKZ and the running time for solving these instances. All the instances except two are run using a single thread on a desktop with a 3.60 GHz Intel Core i7 processor with eight cores and 32 GB 1600 MHz DDR3 memory. The instance $(n = 60, \alpha = 0.005)$ was run on a cluster consisting of 20 c4.8xlarge instances, each having 36 cores with a 60 GB memory (720 threads in total), on the Amazon EC2 platform. The instance $(n = 40, \alpha = 0.02)$ was solved on a cluster consisting of 8 desktops with a 3.60 GHz Intel Core i7 processor with eight cores and 32 GB 1600 MHz DDR3 memory (64 threads in total).

In the experiments we carefully choose the BKZ block size β to ensure the BKZ reduction time is comparable with the enumeration time so as to achieve the reduction on overall running time. Our experiments indeed confirm the folklore that when BKZ reduction time roughly equals that of enumeration time the total running time achieves the minimal. The squared BDD bound R^2 was set as $c \cdot m\alpha^2 q^2$. The successful probability in our pruning strategy is set to be 0.01. From the results of our experiments we find that the relative error rate α plays an important role in the hardness of the LWE problem.

5.2 Comparison with Other Implementations

Recently Kirshanova et al. [18] report a parallel implementation of BDD enumeration for solving LWE. They implement both the Lindner-Peikert [22] nearest planes algorithm and the pruned enumeration method[2] proposed by Liu and Nguyen [21]. They directly implement a pruned parallel enumeration algorithm. Their experiments show that the enumeration algorithm can be nicely parallelized. For example, they achieve a linear speedup by increasing the number of threads even until 10. However, the BKZ reduction is not parallelized. We can observe the impact of Amdahl's Law from their experimental results. For

[2] Kirshanova et al. use linear pruning instead of extreme pruning.

example in order to solve the LWE instance $\mathcal{L}_{80,4093,5}$ their serial implementation needs $4.3 + 13 = 17.3$ h. Their parallel implementation using 10 threads can reduce the enumeration time from 13 h to 1.5 h. Then the total running time is $4.3 + 1.5 = 5.8$ h. That is a 3x speedup by using 10 threads. Even when they increase the number of threads to 20, the total running time is $4.3 + 0.8 = 5.1$ h, which gives a 3.4x speedup by using 20 threads. Although one can circumvent this by using a very small block size for the BKZ reduction part, as we discussed in Sect. 4.4 this choice would increase the total running time of BKZ reduction and pruned enumeration.

On the contrary, our strategy to use extreme pruning and to use many threads working on different basis can scale quite well with respect to the number of threads. Moreover, using extreme pruning can highly reduce the time used by enumeration and thus reduce the total time needed for solving the LWE instance. We compare the running time of our implementation and that of Kirshanova et al. in Table 3. In the table, the time t for BKZ and enumeration stands for the total BKZ time and enumeration time for solving the corresponding LWE instance. In Kirshanova et al.'s setting they fixed the number of samples for the LWE instance and all their experiments use the fixed dimension. We try a different setting where the number of LWE samples are a polynomial of n, say n^2 so that we can use the optimal sub-dimension to reduce the difficulty of the LWE instance.

Table 3. Comparison between Kirshoanova et al. and ours results.

LWE			Kirshanova et al.					Ours implementation					
			Sub-dim	BKZ		Enum		Sub-dim	BKZ		Enum		#Trials
n	q	s	m	β	t	#Treads	t	m	β	t	c	t	
70	4093	6	140	20	65 min	1	44 min	140	20	19 min	1.0	36 s	18
			140	20	65 min	10	5 min	140	15	551 s	1.0	182 s	32
80	4093	5	150	25	4.3 h	1	13 h	150	15	2.8 h	0.8	2.4 h	347
			150	25	1.3 h	10	1.5 h	180	8	6 min	1.0	64 min	12
			150	25	1.3 h	20	50 min	180	10	417 s	1.0	117s	12

The first row of Table 3 shows that extreme pruning can indeed speedup the LWE solver. Kirshanova et al. need 109 min to solve the instance $\mathcal{L}_{70,4093,6}$ on a single thread, while our implementation can solve the instance using the same sub-dimension and block size $\beta = 20$ within 20 min. Further more, their implementation uses 70 min to solve the instance on 10 threads. Since our implementation uses 18 trials to solve the instance we can solve the instance within the time of two rounds if given 10 threads. Basically, we only need less than 4 min to solve the same instance given 10 threads. We note that the block size $\beta = 20$ is not optimal for our implementation. By changing β to 15, we solve the instance $\mathcal{L}_{70,4093,6}$ in 12 min on a single thread.

To further demonstrate the effectiveness of extreme pruning, we compare the performance of both our implementations for the instance $\mathcal{L}_{80,4093,5}$. When we use the same sub-dimension as $m = 150$, the running time of our implementation

on a single thread is 5.2 h which is much smaller than 17.3 h of Kirshanova et al. But the advantage of our implementation lies also in another factor. Notice that the algorithm uses 347 trials to find the correct solution, which means a single trial uses on average only 1 min. Kirshanova et al. solve the instance in more than 2 h using 20 threads. Our implementation is expected to solve the instance in $347/20 = 18$ min. If we have more than 400 threads, we can solve the instance within 1 min. Moreover, if we apply the optimal sub-dimension trick we do not need so many threads to achieve the speedup. For example when we use 180 samples and BKZ block size $\beta = 10$, the total of 12 trials take $417 + 117 = 534$ s. Then with 12 threads our implementation is expected to solve the instance $\mathcal{L}_{80,4093,5}$ using 45 s. On the contrary, the BKZ reduction of Kirshanova et al.'s implementation alone takes 1.3 h.

6 Conclusion and Future Work

This current work described our choice of strategy to solve the BDD problem, namely the details of our implementation and our experimental results on several LWE challenge instances. Our implementation features a embarrassingly parallel design and the use of extreme pruning shows advantages over existing implementations. Potential future work include:

- We choose the optimal BKZ block size β manually in our experiments. This would be impossible for LWE instances with large dimension and/or large relative error rate. Thus it would be useful to explore the relation between the BKZ reduction time and (pruned) enumeration time and use some heuristics to decide the optimal BKZ block size.
- The success probabilities from our experiments seem to be higher than those estimated by Aono's algorithm, thereby resulting in fewer threads. Since we are using parallel implementations of the LWE solver, we have more room for a lower success probability. Lower successful probability can reduce the running time while we can simply add more threads to compensate the low probability of success. In fact our current environment of 20 c4.8xlarge Amazon EC2 instances contains in total more than 700 threads. We can deal with this problem in two ways: first, we can reduce the successful probability for the pruning strategy; second, we can deploy a two-level parallelization by using the first level to run the LWE solver in parallel and using the second level to run the parallel enumeration algorithm.

A Estimate the Cost of BDD Enumeration

Recall that Gaussian heuristic suggests that the enumeration cost at level k of the full enumeration tree is

$$H_k = \frac{V_k(R)}{\prod_{i=m+1-k}^{m} ||\mathbf{b}_i^*||^2}$$

and the total cost is $N = \sum_{k=1}^{m} H_k$.

In the case of BDD enumeration, we set the enumeration bound $R = \sqrt{m}\alpha q$. As for the Gram-Schmidt vector $\mathbf{b}_i^*$, we use the GSA (Geometric Series Assumption) to approximate their lengths.

Geometric Series Assumption (GSA): Schnorr [25] introduced GSA which states that the Gram-Schmidt lengths $||\mathbf{b}_i^*||$ in a BKZ-reduced basis decrease geometrically with quotient r for some constant r related to the reduction algorithm. Specifically $||\mathbf{b}_{i+1}^*||/||\mathbf{b}_i^*|| = r$. Using Gaussian heuristic for the length of the shortest vector we approximate $||\mathbf{b}_1^*||$ as $\delta^m q^{\frac{m-n}{m}}$ where δ is the root Hermite factor achieved by a BKZ reduction algorithm.

Combining Gaussian heuristic and GSA together, we can reformulate Eq. (1) as

$$H_k = \frac{(\sqrt{m}\alpha q)^k \pi^{k/2}}{r^{(2m-k-1)k/2}\delta^{mk} q^{(m-n)k/m}\Gamma(k/2+1)}, \tag{2}$$

where r is the GSA constant, δ is the root Hermite factor achieved by the reduction algorithm and $\Gamma(\cdot)$ is the Gamma function. The total cost (number of nodes in the enumeration tree) is then calculated as

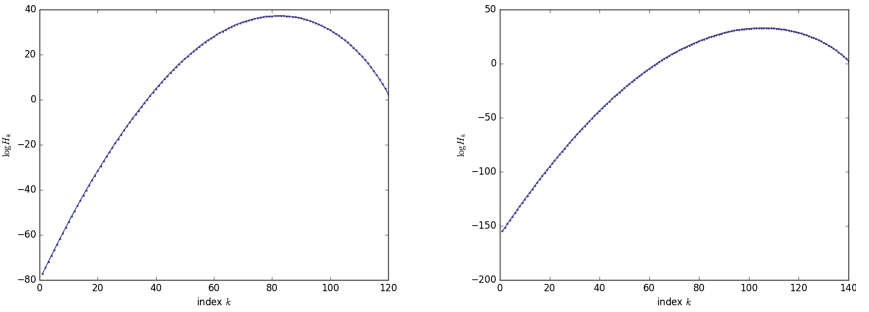
$$N = \sum_{k=1}^{m} H_k. \tag{3}$$

In the BDD case the enumeration cost behaves quite differently from that of the SVP enumeration. In the SVP case, the enumeration radius R is set roughly as $||\mathbf{b}_1^*||$ and the terms H_k reaches maximum when $k = m/2$. But for BDD enumeration, the enumeration radius[3] (bound) is set as $R = \sqrt{m}\alpha q$. This difference actually results in the fact that the cost may decrease as the sub-dimension m increases.

We numerically evaluated the leveled cost H_k and the results are shown in Fig. 4. Figure 4 displays the level cost H_k for different index k for an LWE instance $\mathcal{L}_{50,2503,0.005}$. The basis are reduced by the BKZ-20 reduction algorithm, while the enumeration radius is set as $R = \sqrt{m}\alpha q$. Figure 4a uses sub-dimension $m = 120$ and Fig. 4b uses sub-dimension $m = 140$. From the figures we can observe that the index which maximize H_k is no longer $m/2$ and for the smaller index the value of H_k may be less than 1. We next plot the total cost $N = \sum_{k=1}^{m} H_k$ for an LWE instance by varying the sub-dimension m.

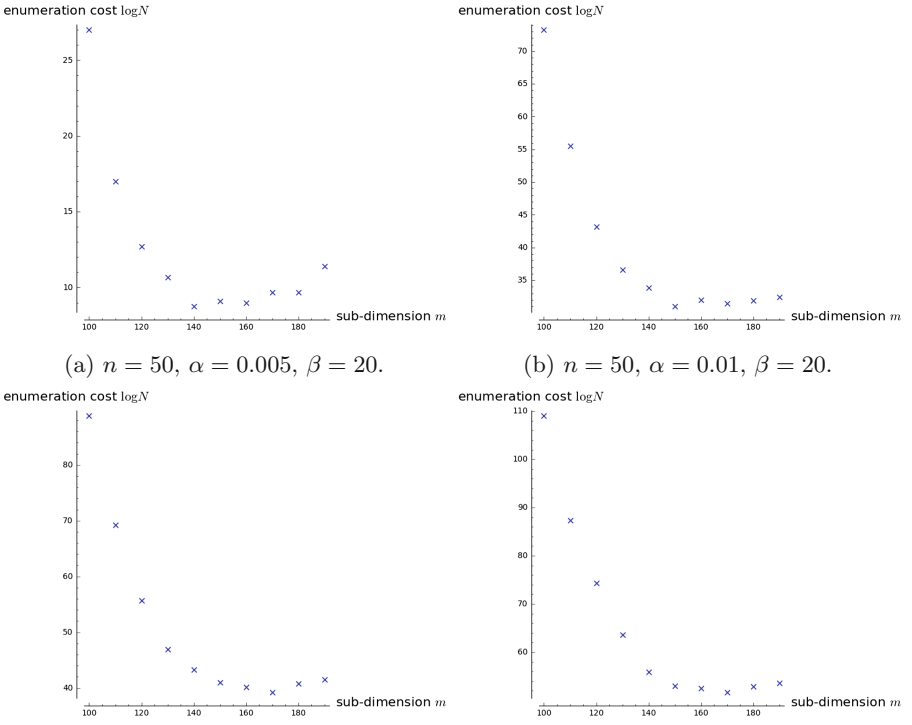
In our experiments, we employ extreme pruning instead of the full enumeration tree. Here, we verify that extreme pruning does not affect the shape of the enumeration cost so that the estimation using full enumeration works effectively for choosing the optimal sub-dimension. Comparing Fig. 5 with Fig. 2, we can see that in the case of enumeration cost with extreme pruning the optimal sub-dimension m is usually a little bigger than that for full enumeration cost. However, the estimated costs do not differ too much. Moreover, when using a larger sub-dimension one has to take the cost of BKZ reduction into consideration. Our

[3] In our implementation we set $R^2 = cm(\alpha q)^2$ for some bound scalar c ranging from 0.8 to 1.3.



(a) $n = 50$, $\alpha = 0.005$, $m = 120$. (b) $n = 50$, $\alpha = 0.005$, $m = 140$.

Fig. 4. Semi-log graph for level cost H_k.



(a) $n = 50$, $\alpha = 0.005$, $\beta = 20$. (b) $n = 50$, $\alpha = 0.01$, $\beta = 20$.

(c) $n = 50$, $\alpha = 0.012$, $\beta = 20$. (d) $n = 50$, $\alpha = 0.015$, $\beta = 20$.

Fig. 5. Semi-log graph for cost N of pruned enumeration. The parameters have the following reference: n is the LWE dimension, α is the relative error rate of the LWE instance and β is the block size for BKZ reduction algorithm used.

experiments in solving the LWE challenge confirms that the estimation using full enumeration cost well serves our purpose to reduce the running time of the whole solving process.

References

1. Ajtai, M.: Generating hard instances of lattice problems. In: Proceedings of 28th Annual ACM Symposium on Theory of Computing, pp. 99–108. ACM (1996)
2. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046 (2015)
3. Albrecht, M.R., Cid, C., Faugere, J., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. Des. Codes Cryptogr. **74**(2), 325–354 (2015)
4. Albrecht, M.R., Cid, C., Faugere, J., Perret, L.: Algebraic algorithms for LWE. Cryptology ePrint Archive, Report 2014/1018 (2014)
5. Albrecht, M.R., Cadé, D., Pujol, X., Stehlé, D.: fplll-4.0, a floating-point LLL implementation. http://perso.ens-lyon.fr/damien.stehle
6. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6755, pp. 403–415. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22006-7_34
7. Aono, Y.: A faster method for computing Gama-Nguyen-Regev's extreme pruning coefficients (2014). arXiv preprint arXiv:1406.0342
8. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of 45th Annual ACM Symposium on Theory of Computing, STOC 2013, pp. 575–584. ACM, New York (2013)
9. Buchmann, J., Büscher, N., Göpfert, F., Katzenbeisser, S., Krämer, J., Micciancio, D., Siim, S., van Vredendaal, C., Walter, M.: Computation, creating cryptographic challenges using multi-party: the LWE challenge. In: Proceedings of 3rd ACM International Workshop on ASIA Public-Key Cryptography (AsiaCCS 2016), pp. 11–20 (2016)
10. Bischof, C., Buchmann, J., Dagdelen, Ö., Fitzpatrick, R., Göpfert, F., Mariano, A.: Nearest planes in practice. In: Ors, B., Preneel, B. (eds.) BalkanCryptSec 2014. LNCS, vol. 9024, pp. 203–215. Springer, Cham (2015). doi:10.1007/978-3-319-21356-9_14
11. Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25385-0_1
12. Detrey, J., Hanrot, G., Pujol, X., Stehlé, D.: Accelerating lattice reduction with FPGAs. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 124–143. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14712-8_8
13. Dagdelen, Ö., Schneider, M.: Parallel enumeration of shortest lattice vectors. In: D'Ambra, P., Guarracino, M., Talia, D. (eds.) Euro-Par 2010. LNCS, vol. 6272, pp. 211–222. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15291-7_21
14. Gama, N., Nguyen, P.Q., Regev, O.: Lattice enumeration using extreme pruning. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 257–278. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_13
15. Gama, N., Nguyen, P.Q.: Predicting Lattice Reduction. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_3

16. Hermans, J., Schneider, M., Buchmann, J., Vercauteren, F., Preneel, B.: Parallel shortest lattice vector enumeration on graphics cards. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 52–68. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12678-9_4
17. Kannan, R.: Minkowski's convex body theorem and integer programming. Math. Oper. Res. **12**(3), 415–440 (1987)
18. Kirshanova, E., May, A., Wiemer, F.: Parallel implementation of BDD enumeration for LWE. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 580–591. Springer, Cham (2016). doi:10.1007/978-3-319-39555-5_31
19. Kuo, P.-C., Schneider, M., Dagdelen, Ö., Reichelt, J., Buchmann, J., Cheng, C.-M., Yang, B.-Y.: Extreme enumeration on GPU and in clouds. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 176–191. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23951-9_12
20. Lenstra, A., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**(4), 515–534 (1982)
21. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: an update. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 293–309. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36095-4_19
22. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19074-2_21
23. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Post-Quantum Cryptography, p. 147 (2009)
24. Regev, O.: On lattices, learning with errors, random linear codes, cryptography. J. ACM (JACM) **56**(6), 34:1–34:40 (2009)
25. Schnorr, C.P.: Lattice reduction by random sampling and birthday methods. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 145–156. Springer, Heidelberg (2003). doi:10.1007/3-540-36494-3_14
26. Schnorr, C.P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Program. **66**(1–3), 181–199 (1994)