Computer Security Symposium 2016
11 - 13 October 2016

An experimental study of the BDD approach for the search LWE problem

Rui Xu[1] Kazuhide Fukushima[1] Shinsaku Kiyomoto[1] Tsuyoshi Takagi[2]

Abstract: Being one of the important variant of post-quantum cryptography, lattice based cryptography has attracted many researches during the last decade. The proved hardness of the Learning With Errors (LWE) problem, assuming the worst case intractability of classic lattice problems such as shortest vector problem and closest vector problem, has made it standard building block in the recent design of lattice based cryptosystem. In this manuscript, we report our implementation of the Bounded Distance Decoding (BDD) approach for solving the search LWE problem. We implemented a parallel version of the pruned enumeration method of the BDD strategy proposed by Liu and Nguyen. The current work aims at providing a thorough understanding of this strategy.
In our implementation we use the embarrassingly parallel design so that the power of mutli-cores can be fully utilized. We let each thread take a randomized basis and enumerate to find the solution independently instead of parallelizing the enumeration algorithm itself. We also carefully design the randomization procedure to make sure no collision occurs among threads. Other optimization includes fine-tuning the BKZ block size and the enumeration bound and pruning coefficients.
The purpose of such experiments is to help understand the practical analysis on search LWE problem and share advice on parameter selection for LWE based cryptosystems. We partially achieve this goal by solving several instances in the TU Darmstadt LWE challenge website. Our current implementation can solve the instance with dimension 60 and relative error rate 0.005 in about 16 days and the instance with dimension 40 and relative error rate 0.02 in about 38 days. To solve this instance we use 20 c4.8xlarge instances, each having 36 cores with a 60 GB memory, on the Amazon EC2 platform.

Keywords: learning with errors, lattice based cryptography, security evaluation

1. Introduction

Decades of development in the lattice-based cryptography area has identified two important primitive hard problems, the Shortest Integer Solution (SIS) problem [1] and the Learning With Errors (LWE) problem [30], to be standard building blocks of modern lattice-based cryptosystems.

The reasons that SIS and LWE become popular in recent development of lattice-based cryptography can be stated as follows:

(1) Perhaps one of the most important reasons is because of their potential usage as post-quantum cryptography, since all current cryptosystems in use today, such as RSA and ElGamal or elliptic curve systems, are all based on the integer factorization problem or discrete logarithm problem which can be solved in probabilistic polynomial time had a quantum computer been available [31].

(2) Either SIS or LWE is simply described as a neat question of solving some linear equation over a finite field. Simple though the description is, these problems are well related to some basic hard lattice problems, such as shortest vector Problems (SVP) and its variants, Bounded Distance Decoding (BDD) problems, which have been attracting researchers for decades while no efficient (probabilistic polynomial) solutions are known [29]. This is true even against a quantum computer, i.e., no quantum algorithm has been found to efficiently solve the hard lattice related problems.

(3) One special property of the above-mentioned lattice hard problems is that they enjoy a worst-case to average-case reduction, which means that the difficulty of a random instance of such problems is as hard as the instances in the worst case [1]. Such a nice property plays a key advantage of basing cryptosystems on lattice problems since randomness lies in the heart of modern cryptography.

(4) Last but no least, SIS and especially LWE has turned out to be an amazingly versatile basis for lattice-based cryptographic constructions. Perhaps the most noticeable of which is the realization of fully homomorphic encryption [9], [17], [18], [32].

1.1 LWE problem

In this work, we focus on the LWE problem proposed by Regev [30]. LWE has attracted more and more atten-

[1] KDDI Research, Inc.
[2] Kyushu University

tion since its proposal. Initially LWE problem was quantumly reduced to the GAPSVP (the decision version of the shortest vector problem) problem or SIVP (Shortest Independent Vector Problem). Which means that LWE is considered hard if there are no algorithms can efficiently solve the GAPSVP or SIVP using a quantum computer. Later, the hardness reduction as been sharpened to accept a classic reduction to these standard lattice problems [10].

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 conversation 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 $\mathbf{a} \in \mathbb{Z}_q^n$ unifromly 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 $\mathcal{L}_{n,q,\mathcal{D}}$. An alternate version of the search LWE problem called decision LWE problem finds more applications in lattice-based cryptographic construction. The decision LWE problem is to distinguish the LWE distribution from uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

1.2 Discrete Gaussian Distribution

We now talk about 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, same as many other previous works regarding LWE we are only concerned with the discrete Gaussian distribution over the ring $\mathbb{Z}_q$ as 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 value of the probability mass function over all integers in each residue class mod q. In the original proposal of Regev, the width parameter is associated with the moduli q as $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$. 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. Preliminaries

2.1 Lattice

A lattice in $\mathbb{R}^m$ is a discrete additive subgroup generated by a (non-unique) basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\}^T$. In another way, we denote the lattice $\Lambda(\mathbf{B})$ generated by $\mathbf{B}$ such that $\Lambda(\mathbf{B}) = \{x | x = \sum_{i=1}^m z_i \mathbf{b}_i\}$, where z_i's are integers. Note that by our convetion, the vector $\mathbf{b}_i$ in the basis matrix $\mathbf{B}$ is its row vector. The rank of 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 call the lattice full rank. A fundamental notion 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 lattice.

$$\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{xA} = 0 \bmod q\},$$

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

It is easy to check that both $\Lambda_q^{\perp}(\mathbf{A})$ and $\Lambda_q(\mathbf{A})$ are q-ary lattices [28]. Note that $\Lambda_q^{\perp}(\mathbf{A})$ and $\Lambda_q(\mathbf{A})$ are dual to each other up to scale. Also notice that the matrix $\mathbf{A}$ is not a basis for these q-ary lattices. However, the lattice basis can be easily calculated via linear algebra. We will detail on how to compute the basis from $\mathbf{A}$ in subsequent section.

2.2 Lattice Reduction

As we have noticed in Section 2.1 that a lattice can be generated from different bases. 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. So 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 get 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. Smaller root-Hermite factor usually implies a reduced basis with higher quality.

Lattice reduction algorithms can be viewed as a case of hierarchical of BKZ [33] based on the parameter blocksize k. The case when $k = 2$ is called LLL reduction, which was invented by Lenstra, Lenstra and Lovasz [25]. LLL reduction is proven to run in polynomial time in the lattice dimension and output a short vector which is within exponential of the minimal length of a lattice Λ, i.e., $\lambda_1(\Lambda)$. When $k = m$, i.e., the full size of the basis, the output basis is HKZ reduced [22] which implies solving the SVP. The situation when k lies in between 2 and m is known as the BKZ-k 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 [13] 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 estimation of the complexity of BKZ.

3. BDD Approach to Solve LWE Search Problem

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. Latter this algorithm was used to analyze the security of code-based cryptosystems. Since LWE can be viewed as a generalization of LPN problem (which generalizes LPN by using more general noise distribution instead of binary noise in LPN), BKW was also adapted to solve LWE by Albrecht et al. [3].

(2) Algebraic approach: Arora-Ge [7] 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. Latter this method was improved by using Grobner 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 detail in the following.

Bounded Distance Decoding (**BDD**): Given m samples $(\mathbf{a_i}, c_i)$ following 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 $\mathbf{c} = \mathbf{As} + \mathbf{e}$, 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 dimension is distributed according to the discrete Gaussian. Consider the q-ary lattice

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

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

There are several ways to solve the **BDD** problem. Lindner and Peikert[27] proposed a variant of Babai's nearest plane algorithm to solve the BDD problem. Bischof et al. [12] and Kirshanova et al. [23] has implemented parallel version of this algorithm and investigated its practical performance. Liu and Nguyen [26] observed that Lindner and Peickert nearest plane algorithm can be viewed as a form of pruned enumeration with pruning strategy different from normal pruning method to bound the projection length but instead to bound the coefficients. Thus 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. Albrecht et al. [6] propose another approach to reduce the BDD problem to unique SVP problem by the embedding technique of Kannan [22].

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 outline the algorithm steps as follows:

- Step 1: Choose an appropriate m number of samples from the LWE samples which will be the dimension of the q-ary lattice in consideration. Denote the subset of LWE samples in matrix representation as $\mathbf{As} + \mathbf{e} = \mathbf{c}$, where $\mathbf{A}$ has size $m \times n$, $\mathbf{s}$ is the n-dimension secret vector we want to find, and $\mathbf{e}$ is the m-dimension error vector chosen from the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}_q,\alpha q}$.
- Step 2: Compute the basis of the q-ary lattice $\Lambda_q(\mathbf{A})$ generated by $\mathbf{A}$, denote it by $\mathbf{B}$. Note that $\mathbf{B}$ is of size $m \times m$.
- Step 3: Randomize the basis $\mathbf{B}$ by multiplying a unimodular matrix $\mathbf{U}$ of size $m \times m$. Denote the randomized basis as $\mathbf{B}'$.
- Step 4: Use BKZ reduction algorithm to reduce the basis $\mathbf{B}'$.
- Step 5: Compute the pruning coefficients and use pruned enumeration to find the closest vector in the lattice $\Lambda_q(\mathbf{A})$ to the vector $\mathbf{c}$ using the BKZ reduced basis $\mathbf{B}'$.
- Step 6: If the closest vector $\mathbf{v}$ is found, solve the LWE secret vector by $\mathbf{s} = \mathbf{A}^{-1}\mathbf{v}$, where $\mathbf{A}^{-1}$ is the inverse of $\mathbf{A}$. Otherwise, goto step (3).

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

4.1 Choose Subdimension

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, one important decision is how to choose 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 $\mathbf{A}$?

The number of samples (i.e., the dimension of the lattice) should be chosen according to the following two considerations:

(1) It should not be too large, otherwise we will be dealing with a lattice of large dimension.

(2) It should not be too small either, for otherwise the sub LWE problem may not have a unique solution.

The first reason seems quite natural. However, during our

experiments we find that the dimension of the lattice does not have a strong impact on the time complexity of solving the LWE problem. For fixed LWE dimension n and relative error rate α, when we increase the number of samples m we do not experience too much of a change in running time. This behavior is what one would expect according to the theory since the hardness of LWE depends on the dimension n and the relative error rate α.

Sometimes a smaller subdimension may even lead to an increased running time. This counter intuitive phenomenon might happen because when the number of samples are two small the corresponding sub LWE problem does not have a unique solution. If we look at the equation $\mathbf{As} + \mathbf{e} = \mathbf{c}$, then 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 precisely, it requires that each element of $\mathbf{e}$ is chosen from the discrete Gaussian distribution with small variant thus can not be too large. TU Darmstat University has held a LWE challenge website[*1] similar to the famous SVP challenge. According to Buchmann et al. [11], 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 subdimension 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. [11] we calculate the probability that the sub LWE problem has more than one solutions. 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 we have two solutions for the secret vector $\mathbf{s}_1$ and $\mathbf{s}_2$ both satisfy the criteria $\mathbf{As}_1 + \mathbf{e}_1 = \mathbf{c} = \mathbf{As}_2 + \mathbf{e}_2$ and $\mathbf{e}_i \leq 2\sqrt{m}\alpha q$. Then by 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 solutions 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 $q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}$. So in our implementation we choose the subdimension m such that the corresponding Gaussian heuristic $q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}$ is larger than $4\sqrt{m}\alpha q$.

4.2 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 vector, 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. Some explanations are as follows. The lattice $\Lambda(\mathbf{A^T})$, which has $\mathbf{A^T}$ as a basis constitutes all lattice vector of $\Lambda_q(\mathbf{A})$ except those belonging to $q\mathbb{Z}^m$. While $q\mathbb{Z}^m$ itself can be viewed as a lattice with the (m dimensional) identity matrix scared 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 lattice $\Lambda(\mathbf{A})$ and $\Lambda(\mathbf{B})$ of the same dimension. Then the basis for the union of $\Lambda(\mathbf{A})$ and $\Lambda(\mathbf{B})$ is $\text{HNF}(\begin{bmatrix}\mathbf{A} \\ \mathbf{B}\end{bmatrix})$.

It is proven that with high probability the basis $\mathbf{B}$ has full rank m (i.e., $\mathbf{B}$ is of size $m \times m$). In our implementation we use a script written in Sage [34] to compute the basis of the q-ary lattice $\Lambda_q(\mathbf{A})$.

4.3 Basis Randomization

Two matrices $\mathbf{B}$ and $\mathbf{B}'$ generates the same lattice if and only if $\mathbf{B}' = \mathbf{UB}$ for some unimodular matrix $\mathbf{U}$. The question of randomizing the lattice basis thus reduces to the question of generating random unimodular matrices. There are many ways to generate unimodular matrices, we choose the following method to generate random unimodular matrices:

(1) Generate two random square matrices $\mathbf{L}$ and $\mathbf{R}$, where $\mathbf{L}$ is a lower triangular matrix with ± 1 as its diagonal elements and where $\mathbf{R}$ is a upper triangular matrix with ± 1 as its diagonal elements. The non-diagonal elements of both matrices are generated randomly uniformly from an interval $[-c, c]$ with c a small positive integer.
(2) Set $\mathbf{U} = \mathbf{LR}$ and output $\mathbf{U}$ as the randomly generated unimodular matrix.

We choose this method to generate unimodular matrices for two reasons:

(1) This method is simple to implement.
(2) When used in multi-thread program we can control the collision of the random matrix in different threads.

We describe how to control the collision in different threads. We assume, for simplicity and without loss of generality, that the diagonal elements of both $\mathbf{L}$ and $\mathbf{R}$ is set to be 1. The following example lists possible constructs for the lower triangle matrix $\mathbf{L}$ and the upper triangle matrix $\mathbf{R}$. In the examples, the symbol 'x' means we do not care what its value is. The argument goes perfectly through even we choose -1 for some of the diagonal elements of the matrices.

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ a_2 & 1 & 0 & \cdots & 0 & 0 \\ a_3 & x & 1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n-1} & x & \cdots & x & 1 & 0 \\ a_n & x & \cdots & \cdots & x & 1 \end{bmatrix}$$

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

$$\mathbf{R} = \begin{bmatrix} 1 & b_2 & b_3 & \cdots & b_{n-1} & b_n \\ 0 & 1 & x & \cdots & x & x \\ 0 & 0 & 1 & \cdots & x & x \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 1 & x \\ 0 & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}$$

$$\mathbf{U} = \mathbf{LR} = \begin{bmatrix} 1 & b_2 & b_3 & \cdots & b_{n-1} & b_n \\ a_2 & x & x & \cdots & x & x \\ a_3 & x & x & \cdots & x & x \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n-1} & x & \cdots & x & x & x \\ a_n & x & \cdots & \cdots & x & x \end{bmatrix}$$

From the example we can see that the first row and first column of the unimodular matrix $\mathbf{U}$ equals the first row of the upper triangle matrix $\mathbf{R}$ and the first column of the lower triangle matrix $\mathbf{L}$, respectively. To make different threads generate the unimodular matrices without collision we can simply fix some elements of the first row of $\mathbf{R}$ and some elements of the first column of $\mathbf{L}$.

Readers might already see from the above example the disadvantage of our method to generate unimodular matrices: the distribution is not uniform! The top-left element of $\mathbf{U}$ is always 1. However as we argue this is no big problem since it is theoretically impossible to generate uniform unimodular matrices simply because there are infinite number of unimodular matrices. One may insist on generating uniform unimodular matrices from a finite subgroup of the unimodular matrices, but as we show that our method already has a large size of randomness so it is not necessary to insist on uniformness. Since the non-diagonal elements of the triangle matrices are generated from $[-c, c]$ and for an $n \times n$ matrix we have $n(n-1)$ such elements. And each element has $2c+1$ possible choices thus the total number of unimodular matrices our method can generate is $(2c+1)^{n(n+1)}$. In our implementation we set $c = 1$ since the randomized lattice basis will have larger entries compared to the origin thus increases the burden of lattice reduction algorithm.

4.4 Basis Reduction

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 k. How to choose an appropriate block size k affects the total running time of our LWE solver. This choice is also related to the parallel strategy we will be using. Parallelism is ubiquitous in today's program design. We have multi-core CPUs even in our laptops. It is natural to implement the LWE solve algorithm in parallel. There are two apparent strategies to us:

(1) Use parallel implementation of enumeration algorithm and parallel implementation of lattice reduction algorithm.

(2) Use sequential implementation of lattice reduction algorithm and enumeration algorithm but instead launch several threads to solve the **BDD** problem using different randomized basis.

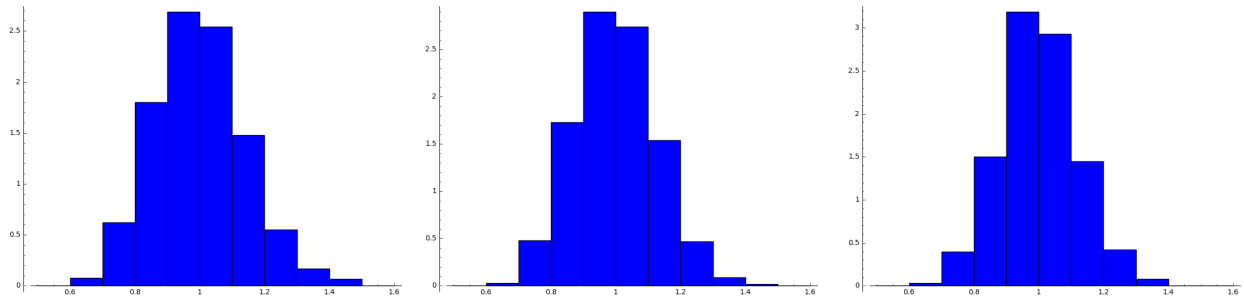
We choose the second design for its simplicity and its embarrassing parallelism. Although there are parallel implementations of lattice enumeration algorithms [15], [16], [21], [24], we do not know any public available implementation of BKZ reduction algorithm. Thus if we want to use parallel implementation 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 $speedup = \frac{1}{1-p}$. When use the combination of BKZ reduction and enumeration to solve the SVP or CVP problem, the common knowledge it that when the running time of BKZ reduction part and the enumeration part are roughly equal, the total running time is minimized. If we want optimal performance then the fraction of parallelizable code would be about 1/2. Then 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, plugging in the parallel enumeration into the BKZ reduction, but those method are either complicate or does not achieve optimal performance gain.

In our implementation we use the embarrassingly parallel design to let each thread work on a different randomized basis, 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 that of the enumeration time.

4.5 Lattice Enumeration

We use extreme pruning [19] for lattice enumeration as suggested by Liu and Nguyen [26]. Given a target vector $\mathbf{t}$, a lattice basis $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_m)$ and a **BDD** radius R, enumeration algorithm enumerates over all lattice vectors $\mathbf{v} \in \mathcal{L}$ such that $||\mathbf{v} - \mathbf{t}|| \leq R$ and find the closest one to solve the **BDD** problem. Gamma et al. [19] suggest using extreme pruning to reduce the number of candidate vectors for enumeration. They also suggest using numerical approximation to generate optimal pruning coefficients by fixing the successful probability and seeking for minimum overhead. Aono [8] also described how to compute the optimal pruning coefficients. We follow Aono's approach to compute the optimal pruning coefficients in our implementation.

The next point is to decide the **BDD** bound R. 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 we can choose $R = 2\sqrt{m}\alpha q$ as the **BDD** bound. However, according to our experiments we find that $2\sqrt{m}\alpha q$ is a little bit too pessimistic. We can actually choose a smaller **BDD** bound R. A smaller **BDD** bound R can dramatically reduce the running time of the enumeration process. To approximate the **BDD** bound R



(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.

we sampled from the according discrete Gaussian distribution and record the squared length of the error vector $\mathbf{e}$. We set squared **BDD** bound R^2 as $c \cdot m\alpha^2 q^2$ for some fixed constant c. The choice of c is made such that $||\mathbf{e}||^2 \leq c \cdot m\alpha^2 q^2$ with overwhelming probability. From our sampling experiments we decide that $c = 1.3$ is an appropriate choice. See Fig. 1 for our experiment results.

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.

5.1 LWE Challenge

TU Darmstadt holds 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 $\mathcal{D}_{\mathbb{Z}_q,\sigma}$.

See Fig.2 for an illustration.

5.2 Our Results

We use our implementation solved several instances from the LWE Challenge website. Please refer to Table 1 for the 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 for the one with parameters $(n = 40, \alpha = 0.02)$ were run on a cluster consisting of 20 c4.8xlarge instances, each having 36 cores with a 60 GB memory, 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 1600MHz DDR3 memory.

In the experiments we carefully choose the BKZ block size k to ensure the BKZ reduction time is comparable with the enumeration time so as to achieve the reduction on overall

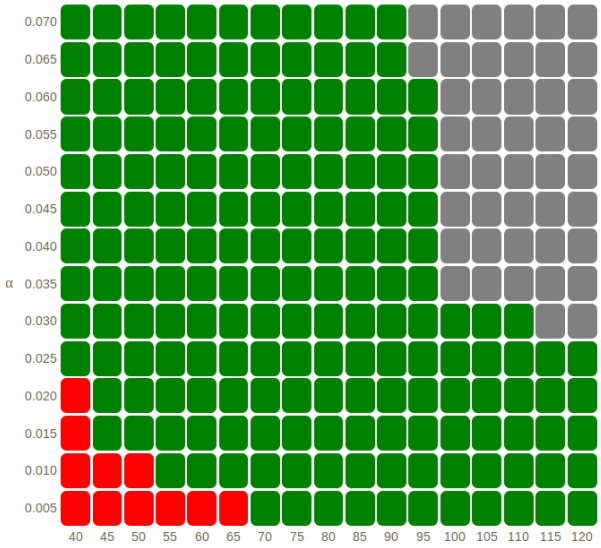


Fig. 2: The LWE challenge. Each cell represents a different LWE instance indexed by the dimension n and relative error rate α. The green cells are unsolved instances and the red ones represent the ones already solved.

running time. And 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 $1.3 \cdot m\alpha^2 q^2$ in all instances and succeeded for all. 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.

6. Future Work

In this preliminary report we just described our choice of strategy to solve the **BDD** problem and details of our implementation and recorded the experiments of solving several LWE challenge instances. There are plenty of areas we can further explore.

- We choose the optimal BKZ block size k manually in our experiment. This would be impossible for LWE instances with large dimension and / or large relative error rate. Thus it would be interesting to explore some estimation of relation between the BKZ reduction time

Table 1: Results on solving some instances from the LWE Challenge website

LWE parameters			BKZ reduction		Enumeration time	Total time
n	α	m	k	time		
40	0.005	140	10	-	-	0.2s
40	0.01	100	30	45	99	144s
40	0.015	120	28	2812s	1937	4749s
40	0.02	140	32	10.9d	27.1d	38d
45	0.005	100	18	4s	1s	5s
45	0.01	120	28	908s	706s	1614s
50	0.005	120	15	26s	22s	48s
55	0.005	140	25	1024s	752s	1776s
60	0.005	140	33	7d	9d	16d

and enumeration time and use some heuristics to decide the optimal BKZ block size.

- Our experiments witnessed the fact that the subdimension does not affect the running time of our LWE solver too much. We will do further experiments to investigate the impact of different choice of subdimensions and understand the reason for this.
- Although we set the successful probability of the pruning strategy to be 0.01, the pruning coefficients we get result in higher successful probability. The impact of this is that we do not need too many repetitions to find the solution. Since we are using parallel running of the LWE solver, we would expect the successful probability to be rather low. 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, while the average rounds needed to find the solution is around 30. We can see that a large part of computing resources is waisted. We can deal with this problem in two ways: firstly, we can reduce the successful probability for the pruning strategy; secondly, we can deploy a two-level parallelism by using the first level to run the LWE solver in parallel and using the second level to run the parallel enumeration algorithm.

References

[1] M. Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108. ACM, 1996.

[2] M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015.

[3] M. R. Albrecht, C. Cid, J. Faugere, R. Fitzpatrick, and L. Perret. On the complexity of the BKW algorithm on LWE. *Des. Codes Cryptography*, 74(2): pages 325-354, 2015.

[4] M. R. Albrecht, C. Cid, J. Faugere and L. Perret. Algebraic algorithms for LWE. Cryptology ePrint Archive, Report 2014/1018, 2014.

[5] M. R. Albrecht, D. Cadé, X. Pujol and D. Stehlé. fplll-4.0, a floating-point LLL implementation. Aailable at `http://perso.ens-lyon.fr/damien.stehle`

[6] M. R. Albrecht, R. Fitzpatrick, R. and F. Göpfert. On the efficacy of solving LWE by reduction to unique-SVP. In *International Conference on Information Security and Cryptology*, pages 293-310, 2014.

[7] S. Arora and R. Ge. New algorithms for learning in presence of errors. In *ICALP 2011, Part 1*, vol. 6755 of *LNCS*, pages 403-415, 2011.

[8] Y. Aono. A faster method for computing Gama-Nguyen-Regev's extreme pruning coefficients. arXiv preprint arXiv:1406.0342 (2014).

[9] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.

[10] Z. Brakerski, A. Langlois, C. Peikert, O. Regev and D. Stehlé. Classical hardness of learning with errors. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 575–584, New York, NY, USA, 2013. ACM.

[11] J. Buchmann, N. Büscher, F. Göpfert, S. Katzenbeisser, J. Krämer, D. Micciancio, S. Siim, C. van Vredendaal and M. Walter. Creating Cryptographic Challenges Using Multi-Party Computation: The LWE Challenge. In *Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography (AsiaCCS 2016)*, pages 11-20, 2016.

[12] C. Bischof, J. Buchmann, O. Dagdelen, R., Fitzpatrick, F. Göpfert and A. Mariano. Nearest Planes in Practice. In *International Conference on Cryptography and Information Security in the Balkans*, pages 203-215, 2015.

[13] Y. M. Chen and P. Q. Nguyen. Bkz 2.0: Better lattice security estimates. In *Advances in Cryptology–ASIACRYPT 2011*, pages 1–20. Springer, 2011.

[14] L. Ducas, A. Durmus, T. Lepoint and V. Lyubashevsky. Lattice Signatures and Bimodal Gaussians. *In Advances in Cryptology – CRYPTO 2013*. Lecture Notes in Computer Science Volume 8042, 2013, pages 40-56.

[15] J. Detrey, G. Hanrot, X. Pujol and D. Stehlé. Accelerating lattice reduction with FPGAs. In *International Conference on Cryptology and Information Security in Latin America*, pages 124-143, 2010.

[16] Ö. Dagdelen and M. Schneider 2010, August. Parallel enumeration of shortest lattice vectors. In *European Conference on Parallel Processing*, pages 211-222, 2010.

[17] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[18] C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology–EUROCRYPT 2012*, pages 465–482. Springer, 2012.

[19] N. Gama, P.Q. Nguyen and O. Regev. Lattice enumeration using extreme pruning. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 257-278, 2010.

[20] C. Gentry, C. Peikert and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. *In Proceedings of the fortieth annual ACM symposium on Theory of computing*. pages 197-206. ACM, 2008.

[21] J. Hermans, M. Schneider, J. Buchmann, F. Vercauteren and B. Preneel. Parallel shortest lattice vector enumeration on graphics cards. In *International Conference on Cryptology in Africa*, pages 52-68, 2010.

[22] R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3): pages 415–440, 1987.

[23] E. Kirshanova, A. May and F. Wiemer. Parallel Implementation of BDD enumeration for LWE. In *International Conference on Applied Cryptography and Network Security*, pages 580-591, 2016.

[24] P.C. Kuo, M. Schneider, . Dagdelen, J. Reichelt, J. Buchmann, C.M. Cheng and B.Y. Yang. Extreme Enumeration on GPU and in Clouds. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 176-191, 2011.

[25] A. Lenstra, H. W. Lenstra, and L. Lovász. Factoring poly-

nomials with rational coefficients. *Mathematische Annalen*, 261(4): pages 515–534, 1982.

[26] M. Liu and P. Q. Nguyen. Solving BDD by enumeration: An update. In *Cryptographers Track at the RSA Conference*, 293-309, 2013.

[27] R. Lindner and C. Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. In *Topics in Cryptology – CT-RSA 2011: The Cryptographers' Track at the RSA Conference*, pages 319–339, San Francisco, CA, USA, 2011. Springer.

[28] D. Micciancio and O. Regev. Lattice-based cryptography. *Post-Quantum Cryptography*, page 147, 2009.

[29] D. Micciancio and S. Goldwasser. Complexity of Lattice Problems: a cryptographic perspective. Kluwer Academic Publishers, 2002.

[30] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6): pages 34:1 - 34:40, 2009.

[31] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM journal on computing*, 26(5): pages 1484–1509, 1997.

[32] M. VanDijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in cryptology–EUROCRYPT 2010*, pages 24–43. Springer, 2010.

[33] C. P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3): pages 181–199, 1994.

[34] *Sage Mathematics Software (Version 6.9)*, The Sage Developers, 2015, `http://www.sagemath.org`