Coded-BKW with Sieving

Qian Guo[1,2], Thomas Johansson[1], Erik Mårtensson[1] and Paul Stankovski[1]

[1] Dept. of Electrical and Information Technology, Lund University[⋆]
[2] ICTEAM/ELEN/Crypto Group, Université Catholique de Louvain[⋆⋆]
{qian.guo,thomas.johansson,erik.martensson,paul.stankovski}@eit.lth.se

Abstract. The Learning with Errors problem (LWE) has become a central topic in recent cryptographic research. In this paper, we present a new solving algorithm combining important ideas from previous work on improving the BKW algorithm and ideas from sieving in lattices. The new algorithm is analyzed and demonstrates an improved asymptotic performance. For Regev parameters $q = n^2$ and noise level $\sigma = n^{1.5}/(\sqrt{2\pi}\log_2^2 n)$, the asymptotic complexity is $2^{0.895n}$ in the standard setting, improving on the previously best known complexity of roughly $2^{0.930n}$. Also for concrete parameter instances, improved performance is indicated.

Keywords: LWE, BKW, Coded-BKW, Lattice codes, Lattice sieving.

1 Introduction

Post-quantum crypto, the area of cryptography in the presence of quantum computers, is currently a major topic in the cryptographic community. Cryptosystems based on hard problems related to lattices are currently intensively investigated, due to their possible resistance against quantum computers. The major problem in this area, upon which cryptographic primitives can be built, is the *Learning with Errors* (LWE) problem.

LWE is an important, efficient and versatile problem. Some famous applications of LWE are to construct Fully Homomorphic encryption schemes [14–16, 21]. A major motivation for using LWE is its connections to lattice problems, linking the difficulty of solving LWE (on average) to the difficulty of solving instances of some (worst-case) famous lattice problems. Let us state the LWE problem.

Definition 1. *Let n be a positive integer, q a prime, and let $\mathcal{X}$ be an error distribution selected as the discrete Gaussian distribution on $\mathbb{Z}_q$. Fix $\mathbf{s}$ to be a secret vector in $\mathbb{Z}_q^n$, chosen according to a uniform distribution. Denote by $L_{\mathbf{s},\mathcal{X}}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing an error $e \in \mathbb{Z}_q$ according to $\mathcal{X}$ and returning*

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

[⋆] Supported by the Swedish Research Council (Grants No. 2015-04528).
[⋆⋆] Supported in part by the European Commission and by the Belgian Fund for Scientific Research through the SECODE project (CHIST-ERA Program).

in $\mathbb{Z}_q^n \times \mathbb{Z}_q$. The (search) LWE *problem is to find the secret vector* $\mathbf{s}$ *given a fixed number of samples from $L_{\mathbf{s},\mathcal{X}}$.*

The definition above gives the *search* LWE problem, as the problem description asks for the recovery of the secret vector $\mathbf{s}$. Another variant is the *decision* LWE problem. In this case the problem is to distinguish between samples drawn from $L_{\mathbf{s},\mathcal{X}}$ and a uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$. Typically, we are then interested in distinguishers with non-negligible advantage.

For the analysis of algorithms solving the LWE problem in previous work, there are essentially two different approaches. One being the approach of calculating the specific number of operations needed to solve a certain instance for a particular algorithm, and comparing specific complexity numbers. The other approach is asymptotic analysis. Solvers for the LWE problem with suitable parameters are expected to have fully exponential complexity, say bounded by 2^{cn} as n tends to infinity. Comparisons between algorithms are made by deriving the coefficient c in the asymptotic complexity expression.

1.1 Related Work

We list the three main approaches for solving the LWE problem in what follows. A good survey with concrete complexity considerations is [6] and for asymptotic comparisons, see [25].

The first class is the algebraic approach, which was initialized by Arora-Ge [8]. This work was further improved by Albrecht et al., using Gröbner bases [2]. Here we point out that this type of attack is mainly, asymptotically, of interest when the noise is very small. For extremely small noise the complexity can be polynomial.

The second and most commonly used approach is to rewrite the LWE problem as a lattice problem, and therefore lattice reduction algorithms [17, 43], sieving and enumeration can be applied. There are several possibilities when it comes to reducing the LWE problem to some hard lattice problem. One is a direct approach, writing up a lattice from the samples and then to treat the search LWE problem as a Bounded Distance Decoding (BDD) problem [34, 35]. One can also reduce the BDD problem to an UNIQUE-SVP problem [5]. Another variant is to consider the distinguishing problem in the dual lattice [37]. Lattice-based algorithms have the advantage of not using an exponential number of samples.

The third approach is the BKW-type algorithms.

BKW variants The BKW algorithm was originally proposed by Blum, Kalai and Wasserman [12] for solving the Learning Parity with Noise (LPN) problem (LWE for $q = 2$). It resembles Wagner's generalized birthday approach [44].

For the LPN case, there have been a number of improvements to the basic BKW approach. In [33], transform techniques were introduced to speed up the search part. Further improvements came in work by Kirchner [28], Bernstein and Tanje [11], Guo et al. [22], Zhang et al. [46], Bogos and Vaudenay [13].

Albrecht et al. were first to apply BKW to the LWE problem [3], which they followed up with Lazy Modulus Switching (LMS) [4], which was further improved by Duc et al. in [20]. The basic BKW approach for LWE was improved in [23] and [29], resulting in an asymptotic improvement. These works improved by reducing a variable number of positions in each step of the BKW procedure as well as introducing a coding approach. Although the two algorithms were slightly different, they perform asymptotically the same and we refer to the approach as coded-BKW. It was proved in [29] that the asymptotic complexity for Regev parameters (public-key cryptography parameter) $q = n^2$ and noise level $\sigma = n^{1.5}/(\sqrt{2\pi}\log_2^2 n)$ is $2^{0.930n}$, the currently best known asymptotic performance for such parameters.

Sieving algorithms A key part of the algorithm to be proposed is the use of sieving in lattices. The first sieving algorithm for solving the shortest vector problem was proposed by Ajtai, Kumar and Sivakumar in [1], showing that SVP can be solved in time and memory $2^{\Theta(n)}$. Subsequently, we have seen the NV-sieve [40], List-sieve [38], and provable improvement of the sieving complexity using the birthday paradox [24, 41].

With heuristic analysis, [40] started to derive a complexity of $2^{0.415n}$, followed by GaussSieve [38], 2-level sieve [45], 3-level sieve [47] and overlattice-sieve [10]. Laarhoven started to improve the lattice sieving algorithms employing algorithmic breakthroughs in solving the nearest neighbor problem, angular LSH [30], and spherical LSH [32]. The asymptotically most efficient approach when it comes to time complexity is Locality Sensitive Filtering (LSF) [9] with both a space and time complexity of $2^{0.292n+o(1)n}$. Using quantum computers, the complexity can be reduced to $2^{0.265n+o(1)n}$ (see [31]) by applying Grover's quantum search algorithm.

1.2 Contributions

We propose a new algorithm for solving the LWE problem combining previous combinatorial methods with an important algorithmic idea – using a sieving approach. Whereas BKW combines vectors to reduce positions to zero, the previously best improvements of BKW, like coded-BKW, reduce more positions but at the price of leaving a small but in general nonzero value in reduced positions. These values are considered as additional noise. As these values increase in magnitude for each step, because we add them together, they have to be very small in the initial steps. This is the reason why in coded-BKW the number of positions reduced in a step is increasing with the step index. We have to start with a small number of reduced positions, in order to not obtain a noise that is too large.

The proposed algorithm tries to solve the problem of the growing noise from the coding part (or LMS) by using a sieving step to make sure that the noise from treated positions does not grow, but stays approximately of the same size. The basic form of the new algorithm then contains two parts in each iterative step. The first part reduces the magnitude of some particular positions by finding pairs of vectors that can be combined. The second part performs a sieving step covering all positions from all previous steps, making sure that the magnitude of the resulting vector components is roughly as in the already size-reduced part of the incoming vectors.

We analyze the new algorithm from an asymptotic perspective, proving a new improved asymptotic performance. Again, for asymptotic Regev parameters $q = n^2$ and noise level $\sigma = n^{1.5}$, the result is a time and space complexity of $2^{0.895n}$, which is a significant asymptotic improvement. We also get a first quantum acceleration for Regev's parameter setting by using the performance of sieving in the quantum setting. We additionally sketch on an algorithmic description for concrete instances. Here we can use additional non-asymptotic improvements and derive approximate actual complexities for specific instances, demonstrating its improved performance also for concrete instances.

1.3 Organization

The remaining parts of the paper are organized as followed. We start with some preliminaries in Section 2, including more basics on LWE, discrete Guassians and sieving in lattices. In Section 3 we review the details of the BKW algorithm and some recent improvements. Section 4 just contains a simple reformulation. In Section 5 we give the new algorithm in its basic form and in Section 6 we derive the optimal parameter selection and perform the asymptotic analysis. In Section 7 we briefly overview how the algorithm could be extended in the non-asymptotic case, including additional non-asymptotic improvements. We compute some rough estimates on expected number of operations to solve specific instances, indicating improved performance in comparison with previous complexity results. We conclude the paper in Section 8.

2 Background

2.1 Notations

Throughout the paper, the following notations are used.

- We write $\log(\cdot)$ for the base 2 logarithm and $\ln(\cdot)$ for the natural logarithm.
- In an n-dimensional Euclidean space $\mathbb{R}^n$, by the norm of a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ we refer to its L_2-norm, defined as

$$\|\mathbf{x}\| = \sqrt{x_1^2 + \cdots + x_n^2}.$$

  We then define the Euclidean distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$ as $\|\mathbf{x} - \mathbf{y}\|$.
- For an $[N, K]$ linear code, N denotes the code length and K denotes the dimension.

2.2 LWE Problem Description

Rather than giving a more formal definition of the decision version of LWE, we instead reformulate the search LWE problem, because our main purpose is to investigate its solving complexity. Assume that m samples

$$(\mathbf{a}_1, z_1), (\mathbf{a}_2, z_2), \ldots, (\mathbf{a}_m, z_m),$$

are drawn from the LWE distribution $L_{\mathbf{s},\mathcal{X}}$, where $\mathbf{a}_i \in \mathbb{Z}_q^n, z_i \in \mathbb{Z}_q$. Let $\mathbf{z} = (z_1, z_2, \ldots, z_m)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_m) = \mathbf{sA}$. We can then write

$$\mathbf{z} = \mathbf{sA} + \mathbf{e},$$

where $\mathbf{A} = \left[\mathbf{a}_1^{\mathrm{T}}\ \mathbf{a}_2^{\mathrm{T}} \cdots \mathbf{a}_n^{\mathrm{T}}\right]$, $z_i = y_i + e_i = \langle \mathbf{s}, \mathbf{a}_i \rangle + e_i$ and $e_i \xleftarrow{\$} \mathcal{X}$. Therefore, we have reformulated the search LWE problem as a decoding problem, in which the matrix $\mathbf{A}$ serves as the generator matrix for a linear code over $\mathbb{Z}_q$ and $\mathbf{z}$ is the received word. We see that the problem of searching for the secret vector $\mathbf{s}$ is equivalent to that of finding the codeword $\mathbf{y} = \mathbf{sA}$ such that the Euclidean distance $||\mathbf{y} - \mathbf{z}||$ is minimal.

The Secret-Noise Transformation An important transformation [7, 28] can be applied to ensure that the secret vector follows the same distribution $\mathcal{X}$ as the noise. The procedure works as follows. We first write $\mathbf{A}$ in systematic form via Gaussian elimination. Assume that the first n columns are linearly independent and form the matrix $\mathbf{A_0}$. We then define $\mathbf{D} = \mathbf{A_0}^{-1}$ and write $\hat{\mathbf{s}} = \mathbf{sD}^{-1} - (z_1, z_2, \ldots, z_n)$. Hence, we can derive an equivalent problem described by $\hat{\mathbf{A}} = (\mathbf{I}, \hat{\mathbf{a}}_{n+1}^{\mathrm{T}}, \hat{\mathbf{a}}_{n+2}^{\mathrm{T}}, \cdots, \hat{\mathbf{a}}_m^{\mathrm{T}})$, where $\hat{\mathbf{A}} = \mathbf{DA}$. We compute

$$\hat{\mathbf{z}} = \mathbf{z} - (z_1, z_2, \ldots, z_n)\hat{\mathbf{A}} = (\mathbf{0}, \hat{z}_{n+1}, \hat{z}_{n+2}, \ldots, \hat{z}_m).$$

Using this transformation, one can assume that each entry in the secret vector is now distributed according to $\mathcal{X}$.

The noise distribution $\mathcal{X}$ is usually chosen as the discrete Gaussian distribution, which will be briefly discussed in Section 2.3.

2.3 Discrete Gaussian Distribution

We start by defining the discrete Gaussian distribution over $\mathbb{Z}$ with mean 0 and variance σ^2, denoted $D_{\mathbb{Z},\sigma}$. That is, the probability distribution obtained by assigning a probability proportional to $\exp(-x^2/2\sigma^2)$ to each $x \in \mathbb{Z}$. Then the discrete Gaussian distribution $\mathcal{X}$ over $\mathbb{Z}_q$ with variance σ^2 (also denoted $\mathcal{X}_\sigma$) can be defined by folding $D_{\mathbb{Z},\sigma}$ and accumulating the value of the probability mass function over all integers in each residue class modulo q.

Following the path of previous work [3], we assume that in our discussed instances, the discrete Gaussian distribution can be approximated by the continuous counterpart. For instance, if X is drawn from $\mathcal{X}_{\sigma_1}$ and Y is drawn from $\mathcal{X}_{\sigma_2}$, then $X + Y$ is regarded as being drawn from $\mathcal{X}_{\sqrt{\sigma_1^2+\sigma_2^2}}$. This approximation is widely adopted in literature.

The sample complexity for distinguishing. To estimate the solving complexity, we need to determine the number of required samples to distinguish between the uniform distribution on $\mathbb{Z}_q$ and $\mathcal{X}_\sigma$. Relying on standard theory from statistics, using either previous work [34] or Bleichenbacher's definition of bias [39], we can conclude that the required number of samples is

$$\mathcal{O}\left(e^{2\pi\left(\frac{\sigma\sqrt{2\pi}}{q}\right)^2}\right).$$

2.4 Sieving in Lattices

We here give a brief introduction to the sieving idea and its application in lattices for solving the shortest vector problem (SVP). For an introduction to lattices, the SVP problem, and sieving algorithms, see e.g. [9].

In sieving, we start with a list $\mathcal{L}$ of relatively short lattice vectors. If the list size is large enough, we will obtain many pairs of $\mathbf{v}, \mathbf{w} \in \mathcal{L}$, such that $\|\mathbf{v} \pm \mathbf{w}\| \leq \max\{\|\mathbf{v}\|, \|\mathbf{w}\|\}$. After reducing the size of these lattice vectors a polynomial number of times, one can expect to find the shortest vector.

The core of sieving is thus to find a close enough neighbor $\mathbf{v} \in \mathcal{L}$ efficiently, for a vector $\mathbf{w} \in \mathcal{L}$, thereby reducing the size by further operations like addition or subtraction. This is also true for our newly proposed algorithm in a later section, since by sieving we solely desire to control the size of the added/subtracted vectors. For this specific purpose, many famous probabilistic algorithms have been proposed, e.g., Locality Sensitive Hashing (LSH) [27], Bucketing coding [19], May-Ozerov's algorithm [36] in the Hamming metric with important applications to decoding binary linear codes.

In the Euclidean metric, the state-of-the-art algorithm in the asymptotic sense is Locality Sensitive Filtering (LSF) [9], which requires $2^{0.2075n+o(n)}$ samples. In the classic setting, the time and memory requirements are both in the order of $2^{0.292n+o(n)}$. The constant hidden in the running time exponent can be reduced to 0.265 in the scenario of quantum computing. In the remaining part of the paper, we choose the LSF algorithm for the best asymptotic performance when we need to instantiate the sieving method.

3 The BKW Algorithm

The BKW algorithm is the first sub-exponential algorithm for solving the LPN problem, originally proposed by Blum, Kalai and Wasserman [12]. It can also be trivially adopted to the LWE problem, with single-exponential complexity.

3.1 Plain BKW

The algorithm consists of two phases: the reduction phase and the solving phase. The essential improvement comes from the first phase, whose underlying fundamental idea is the same as Wagner's generalized birthday algorithm [44]. That is, using an iterative collision procedure on the columns in the matrix $\mathbf{A}$, one can reduce its row dimension step by step, and finally reach a new LWE instance with a much smaller dimension. The solving phase can then be applied to recover the secret vector. We describe the core procedure of the reduction phase, called a plain BKW step, as follows. Let us start with $\mathbf{A}_0 = \mathbf{A}$.

Dimension reduction: In the i-th iteration, we look for combinations of two columns in $\mathbf{A}_{i-1}$ that add (or subtract) to zero in the last b entries. Suppose that one finds two columns $\mathbf{a}_{j_1,i-1}^{\mathrm{T}}, \mathbf{a}_{j_2,i-1}^{\mathrm{T}}$ such that

$$\mathbf{a}_{j_1,i-1} \pm \mathbf{a}_{j_2,i-1} = [* * \cdots * \underbrace{0\, 0 \cdots 0}_{b \text{ symbols}}],$$

where $*$ means any value. We then generate a new vector $\mathbf{a}_{j,i} = \mathbf{a}_{j_1,i-1} \pm \mathbf{a}_{j_2,i-1}$. We obtain a new generator matrix $\mathbf{A}_i$ for the next iteration, with its dimension reduced by b, if we remove the last b all-zero positions with no impact on the output of the inner product operation. We also derive a new "observed symbol" as $z_{j,i} = z_{j_1,i-1} \pm z_{j_2,i-1}$.

A trade-off: After one step of this procedure, we can see that the new noise variable $e_{j,i} = e_{j_1,i-1} \pm e_{j_2,i-1}$. If the noise variables $e_{j_1,i-1}$ and $e_{j_2,i-1}$ both follow the Gaussian distribution with variance σ_{i-1}^2, then the new noise variable $e_{j,i}$ is considered Gaussian distributed with variance $\sigma_i^2 = 2\sigma_{i-1}^2$.

After t_0 iterations, we have reduced the dimension of the problem to $n - t_0 b$. The final noise variable is thus a summation of 2^{t_0} noise variables generated from the LWE oracle. We therefore know that the noise connected to each column is of the form

$$e = \sum_{j=1}^{2^{t_0}} e_{i_j},$$

and the total noise is approximately Gaussian with variance $2^{t_0} \cdot \sigma^2$.

The remaining solving phase is to solve this transformed LWE instance. This phase does not affect its asymptotic complexity but has significant impact on its actual running time for concrete instances.

Similar to the original proposal [12] for solving LPN, which recovers 1 bit in the secret vector via majority voting, Albrecht et al. [3] exhaust one secret entry using a distinguisher. The complexity is further reduced by Duc et al. [20] using Fast Fourier Transform (FFT) to recover several secret entries simultaneously.

3.2 Coded-BKW

As described above, in each BKW step, we try to collide a large number of vectors $\mathbf{a}_i$ in a set of positions denoted by an index set I. We denote this subvector of a vector $\mathbf{a}$ as a $\mathbf{a}_I$. We set the size of the collision set to be $(\frac{q^b-1}{2})$, a very important parameter indicating the final complexity of the algorithm.

In this part we describe another idea that, instead of zeroing out the vector $\mathbf{a}_I$ by collisions, we try to collide vectors to make $\mathbf{a}_I$ small. The advantage of this idea is that one can handle more positions in one step for the same size of the collision set.

This idea was first formulated by Albrecht et al. in PKC 2014 [4], aiming for solving the LWE problem with a small secret. They proposed a new technique called Lazy Modulus Switching (LMS). Then, in CRYPTO 2015, two new algorithms with similar underlying algorithmic ideas were proposed independently in [23] and [29], highly enhancing the performance in the sense of both asymptotic and concrete complexity. Using the secret-noise transformation, these new algorithms can be used to solve the standard LWE problem.

In this part we use the notation from [23] to describe the BKW variant called coded-BKW, as it has the best concrete performance, i.e., it can reduce the magnitude of the noise by a constant factor compared with its counterpart technique LMS.

The core step – the coded-BKW step – can be described as follows.

Considering step i in the reduction phase, we choose a q-ary $[n_i, b]$ linear code, denoted $\mathcal{C}_i$, that can be employed to construct a lattice code, e.g., using Construction A (see [18] for details). The sub-vector $\mathbf{a}_I$ can then be written in terms of its two constituents, the codeword part $\mathbf{c}_I \in \mathcal{C}_i$ and an error part $\mathbf{e}_I \in \mathbb{Z}_q^{N_i}$. That is,

$$\mathbf{a}_I = \mathbf{c}_I + \mathbf{e}_I. \tag{1}$$

We rewrite the inner product $\langle \mathbf{s}_I, \mathbf{a}_I \rangle$ as

$$\langle \mathbf{s}_I, \mathbf{a}_I \rangle = \langle \mathbf{s}_I, \mathbf{c}_I \rangle + \langle \mathbf{s}_I, \mathbf{e}_I \rangle .$$

We can cancel out the part $\langle \mathbf{s}_I, \mathbf{c}_I \rangle$ by subtracting two vectors mapped to the same codeword, and the remaining difference is the noise. Due to symmetry, the size of the collision set can be $\frac{q^b-1}{2}$, as in the plain BKW step.

If we remove n_i positions in the i-th step, then we have removed $\sum_{i=1}^{t} n_i$ positions ($n_i \geq b$) in total. Thus, after guessing the remaining secret symbols in the solving phase, we need to distinguish between the uniform distribution and the distribution representing a sum of noise variables, i.e.,

$$z = \sum_{j=1}^{2^t} e_{i_j} + \sum_{i=1}^{n} s_i (E_i^{(1)} + E_i^{(2)} + \cdots + E_i^{(t)}), \tag{2}$$

where $E_i^{(h)} = \sum_{j=1}^{2^{t-h+1}} \hat{e}_{i_j}^{(h)}$ and $\hat{e}_{i_j}^{(h)}$ is the noise introduced in the h-th coded-BKW step. Here at most one error term $E_i^{(h)}$ is non-zero for one position in the index set, and the overall noise can be estimated according to Equation (2).

The remaining problem is to analyze the noise level introduced by coding. In [23], it is assumed that every $E_i^{(h)}$ is close to a Gaussian distribution, which is tested in implementation. Based on known results on lattice codes, the standard deviation σ introduced by employing a q-ary $[N, k]$ linear code is estimated by

$$\sigma \approx q^{1-k/N} \cdot \sqrt{G(\Lambda_{N,k})}, \tag{3}$$

where $G(\Lambda_{N,k})$ is a code-related parameter satisfying

$$\frac{1}{2\pi e} < G(\Lambda_{N,k}) \leq \frac{1}{12}.$$

In [23], the chosen codes are with varying rates to ensure that the noise contribution of each position is equal. This is principally similar to the operation of changing the modulus size in each reduction step in [29].

4 A Reformulation

Let us reformulate the LWE problem and the steps in the different algorithms in a matrix form. Recall that we have the LWE samples in the form $\mathbf{z} = \mathbf{s}\mathbf{A} + \mathbf{e}$. We write this as

$$(\mathbf{s}, \mathbf{e}) \begin{pmatrix} \mathbf{A} \\ \mathbf{I} \end{pmatrix} = \mathbf{z}. \tag{4}$$

The entries in the unknown left-hand side vector $(\mathbf{s}, \mathbf{e})$ are all i.i.d. The matrix above is denoted as $\mathbf{H_0} = \begin{pmatrix} \mathbf{A} \\ \mathbf{I} \end{pmatrix}$ and it is a known quantity, as well as $\mathbf{z}$.

By multiplying Equation (4) from the right with special matrices $\mathbf{P_i}$ we are going to reduce the size of columns in the matrix. Starting with

$$(\mathbf{s}, \mathbf{e})\mathbf{H_0} = \mathbf{z},$$

we find a matrix $\mathbf{P_0}$ and form $\mathbf{H_1} = \mathbf{H_0}\mathbf{P_0}$, $\mathbf{z}_1 = \mathbf{z}\mathbf{P_0}$, resulting in

$$(\mathbf{s}, \mathbf{e})\mathbf{H_1} = \mathbf{z}_1.$$

Continuing this process for t steps, we have formed $\mathbf{H_t} = \mathbf{H_0}\mathbf{P_0} \cdots \mathbf{P_{t-1}}$, $\mathbf{z}_t = \mathbf{z}\mathbf{P_0} \cdots \mathbf{P_{t-1}}$.

Plain BKW can be described as each $\mathbf{P_i}$ having columns with only two nonzero entries, both from the set $\{-1, 1\}$. The BKW procedure subsequently cancels rows in the $\mathbf{H_i}$ matrices in a way such that $\mathbf{H_t} = \begin{pmatrix} \mathbf{0} \\ \mathbf{H'_t} \end{pmatrix}$, where columns of $\mathbf{H'_t}$ have 2^t non-zero entries[3]. The goal is to minimize the magnitude of the column entries in $\mathbf{H_t}$. The smaller magnitude, the larger advantage in the corresponding samples.

The improved techniques like LMS and coded-BKW reduce the $\mathbf{H_t}$ similar to the BKW, but improves by using the fact that the top rows of $\mathbf{H'_t}$ do not have to be canceled to $\mathbf{0}$. Instead, entries are allowed to be of the same norm as in the $\mathbf{H'_t}$ matrix.

[3] Sometimes we get a little fewer than 2^t entries since 1s can overlap. However, this probability is low and does not change the analysis.

5 A BKW-Sieving Algorithm for the LWE Problem

The algorithm we propose uses a similar structure as the coded-BKW algorithm. The new idea involves changing the BKW step to also include a sieving step. In this section we give the algorithm in a simple form, allowing for some asymptotic analysis. We exclude some steps that give non-asymptotic improvements. These are considered in a later section. We assume that each entry in the secret vector $\mathbf{s}$ is distributed according to $\mathcal{X}$.

5.1 Initial Guessing Step

We select a few entries of $\mathbf{s}$ and guess these values (according to $\mathcal{X}$). We run through all likely values and for each of them we do the steps below. Based on a particular guess, the sample equations need to be rewritten accordingly.

For simplicity, the remaining unknown values are still denoted $\mathbf{s}$ after this guessing step and the length of $\mathbf{s}$ is still denoted n.

5.2 Transformation Steps

We start with some simplifying notation. The n positions in columns in $\mathbf{A}$ (first n positions in columns of $\mathbf{H}$) are considered as a concatenation of smaller vectors. We assume that these vectors have lengths which are $n_1, n_2, n_3, \ldots, n_t$, in such a way that $\sum_{i=1}^{t} n_i = n$. Also, let $N_j = \sum_{i=1}^{j} n_i$, for $j = 1, 2, \ldots, t$.

Before explaining the algorithmic steps, we introduce two notions that will be used later.

Notation CodeMap$(\mathbf{h}, i)$**:** We assume, following the idea of Coded-BKW, that we have fixed a lattice code $\mathcal{C}_i$ of length n_i. The vector $\mathbf{h}$ fed as input to CodeMap is first considered only restricted to the positions $N_{i-1}+1$ to N_i, i.e., as a vector of length n_i. This vector, denoted $\mathbf{h}_{[N_{i-1}+1, N_i]}$ is then mapped to the closest codeword in $\mathcal{C}_i$. This closest codeword is denoted CodeMap$(\mathbf{h}, i)$.

The code $\mathcal{C}_i$ needs to have an associated procedure of quickly finding the closest codeword for any given vector. One could then use a simple code or a more advanced code. From an asymptotic viewpoint, it does not matter, but in a practical implementation there can be a difference. We are going to select the parameters in such a way that the distance to the closest codeword is expected to be no more than $\sqrt{n_i} \cdot B$, where B is a constant.

Notation Sieve$(\mathcal{L}_\Delta, i, \sqrt{N_i} \cdot B)$**:** The input $\mathcal{L}_\Delta$ contains a list of vectors. We are only considering them restricted to the first N_i positions. This procedure will find all differences between any two vectors such that the norm of the difference restricted to the first N_i positions is less than $\sqrt{N_i} \cdot B$. All such differences are put in a list $\mathcal{S}_\Delta$ which is the output of the procedure.

We assume that the vectors in the list $\mathcal{L}_\Delta$ restricted to the first N_i positions, all have a norm of about $\sqrt{N_i} \cdot B$. Then the problem is solved by algorithms for sieving in lattices, for example using Locality-Sensitive Hashing/Filtering.

For the description of the main algorithm, recall that

$$(\mathbf{s}, \mathbf{e})\mathbf{H_0} = \mathbf{z},$$

where $\mathbf{H_0} = \begin{pmatrix} \mathbf{A} \\ \mathbf{I} \end{pmatrix}$. We are going to perform t steps to transform $\mathbf{H_0}$ into $\mathbf{H_t}$ such that the columns in $\mathbf{H_t}$ are "small". Again, we look at the first n positions in a column corresponding to the $\mathbf{A}$ matrix. Since we are only adding or subtracting columns using coefficients in $\{-1, 1\}$, the remaining positions in the column are assumed to contain 2^i nonzero positions either containing a -1 or a 1, after i steps.

5.3 A BKW-Sieving Step

We are now going to fix an average level of "smallness" for a position, which is a constant denoted B, as above. The idea of the algorithm is to keep the norm of considered vectors of some length n' below $\sqrt{n'} \cdot B$.

A column $\mathbf{h} \in \mathbf{H_0}$ will now be processed by first computing $\Delta = \text{CodeMap}(\mathbf{h}, 1)$. Then we place $\mathbf{h}$ in the list $\mathcal{L}_\Delta$. After running through all columns $\mathbf{h} \in \mathbf{H_0}$ they have been sorted into K lists $\mathcal{L}_\Delta$.

We then run through the lists, each containing roughly m/K columns. We perform a sieving step, according to $\mathcal{S}_\Delta = Sieve(\mathcal{L}_\Delta, \sqrt{N_1} \cdot B)$. The result is a list of vectors, where the norm of each vector restricted to the first N_1 positions is less than $\sqrt{N_1} \cdot B$. The indices of any two columns, i_j, i_k are kept in such a way that we can compute a new received symbol $z = z_{i_j} - z_{i_k}$. All vectors in all lists $\mathcal{S}_\Delta$ are now put as columns in $\mathbf{H_1}$. We now have a matrix $\mathbf{H_1}$ where the norm of each column restricted to the first n_1 positions is less than $\sqrt{N_1} \cdot B$. This is the end of the first step.

Next, we repeat roughly the same procedure another $t-1$ times. A column $\mathbf{h} \in \mathbf{H_{i-1}}$ will now be processed by first computing $\Delta = \text{CodeMap}(\mathbf{h}, i)$. We place $\mathbf{h}$ in the list $\mathcal{L}_\Delta$. After running through all columns $\mathbf{h} \in \mathbf{H_{i-1}}$ they have been sorted in K lists $\mathcal{L}_\Delta$.

We run through all lists, where each list contains roughly m/K columns. We perform a sieving step, according to $\mathcal{S}_\Delta = \text{Sieve}(\mathcal{L}_\Delta, i, \sqrt{N_i} \cdot B)$. The result is a list of vectors where the norm of each vector restricted to the first N_i positions is less than $\sqrt{N_i} \cdot B$. A new received symbol is computed. All vectors in all lists $\mathcal{S}_\Delta$ are now put as columns in $\mathbf{H_i}$. We get a matrix $\mathbf{H_i}$ where the norm of each column restricted to the first N_i positions is less than $\sqrt{N_i} \cdot B$. This is repeated for $i = 2, \ldots, t$. We assume that the parameters have been chosen in such a way that each matrix $\mathbf{H_i}$ can have m columns.

After performing these t steps we end up with a matrix $\mathbf{H_t}$ such that the norm of columns restricted to the first n positions is bounded by $\sqrt{n} \cdot B$ and the norm of the last m positions is roughly $2^{t/2}$. Altogether, this should result in samples generated as

$$\mathbf{z} = (\mathbf{s}, \mathbf{e})\mathbf{H_t}.$$

The values in the $\mathbf{z}$ vector are then roughly Gaussian distributed, with variance $\sigma^2 \cdot (nB^2 + 2^t)$. By running a distinguisher on the created samples $\mathbf{z}$ we can verify whether our initial guess is correct or not. After restoring some secret value, the whole procedure can be repeated, but for a smaller dimension.

5.4 Algorithm Summary

Algorithm 1 Coded-BKW with Sieving (main steps)

Input: Matrix $\mathbf{A}$ with n rows and m columns, received vector $\mathbf{z}$ of length m and algorithm parameters $t, n_i, 1 \leq i \leq t$, B

```
change the distribution of the secret vector (Gaussian elimination)
for $i$ from 1 to $t$ do:
    for all columns $\mathbf{h} \in \mathbf{H_{i-1}}$ do:
        $\Delta = \text{CodeMap}(\mathbf{h}, i)$
        put $\mathbf{h}$ in list $\mathcal{L}_\Delta$
    for all lists $\mathcal{L}_\Delta$ do:
        $\mathcal{S}_\Delta = \text{Sieve}(\mathcal{L}_\Delta, i, \sqrt{N_i} \cdot B)$
        put all $\mathcal{S}_\Delta$ as columns in $\mathbf{H_i}$
exhaustively guess the $\mathbf{s}_n$ entry using hypothesis testing
```

A summary of coded-BKW with sieving is contained in Algorithm 1.

Note that one may also use some advanced distinguisher, e.g., the FFT distinguisher, which is important to the concrete complexity but not the asymptotic performance.

5.5 High-Level Comparison with Previous BKW Versions

A high-level comparison between the behaviors of plain BKW, coded-BKW and coded-BKW with sieving is shown in Figure 1.

Initially the average norm of all elements in a sample vector $\mathbf{a}$ is around $q/2$, represented by the first row in the figure. Plain BKW then gradually works towards a zero vector by adding/subtracting vectors in each step such that a fixed number of positions get canceled out to 0.

The idea of coded-BKW is to not cancel out the positions completely, and thereby allowing for longer steps. The positions that are not canceled out increase in magnitude by a factor of $\sqrt{2}$ in each step. To end up with an evenly distributed noise vector in the end we can let the noise in the new almost canceled positions increase by a factor of $\sqrt{2}$ in each step. Thus we can gradually increase the step size.

When reducing positions in coded-BKW, the previously reduced positions increase in magnitude by a factor of $\sqrt{2}$. However, the sieving step in coded-BKW with sieving makes sure that the previously reduced positions do not increase in magnitude. Thus, initially, we do not have to reduce the positions as much as in coded-BKW. However, the sieving process gets more expensive the more positions we work with, and we must therefore gradually decrease the step size.

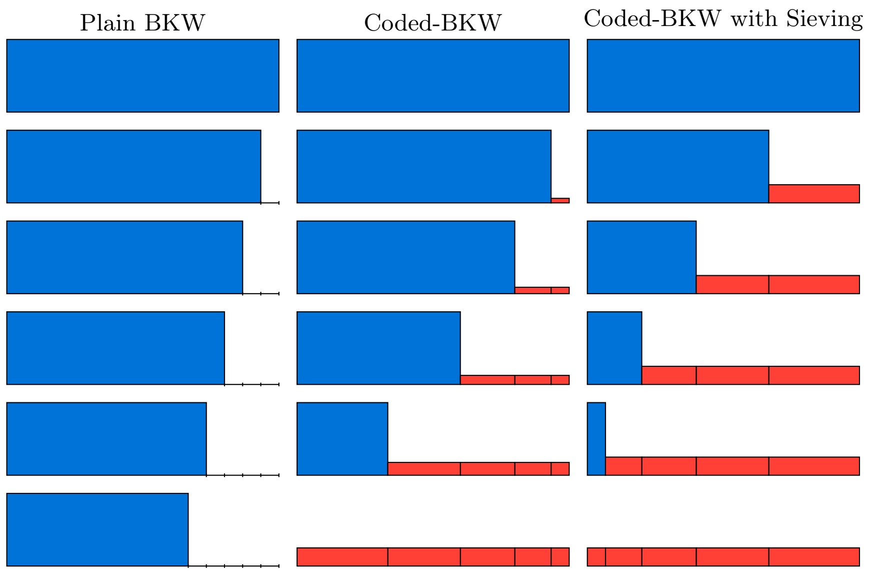

Fig. 1. A high level explanation of how the different versions of the BKW algorithm work.

6 Parameter Selection and Asymptotic Analysis

After each step, positions that already have been treated should remain at some given magnitude B. That is, the average (absolute) value of a treated position should be very close to B. This property is maintained by the way in which we apply the sieving part at each reduction step. After t steps we have therefore produced vectors of average norm $\sqrt{n} \cdot B$.

Assigning the number of samples to be $m = 2^k$, where 2^k is a parameter that will decide the total complexity of the algorithm, we will end up with roughly $m = 2^k$ samples after t steps. As already stated, these received samples will be roughly Gaussian with variance $\sigma^2 \cdot (nB^2 + 2^t)$. We assume that the best strategy is to keep the magnitudes of the two different contributions of the same order, so we choose $nB^2 \approx 2^t$.

Furthermore, in order to be able to recover a single secret position using m samples, we need

$$m = \mathcal{O}\left(e^{4\pi^2 \cdot \frac{\sigma^2 \cdot (nB^2 + 2^t)}{q^2}}\right).$$

Thus, we have

$$\ln 2 \cdot k = 4\pi^2 \cdot \frac{\sigma^2 \cdot (nB^2 + 2^t)}{q^2},$$

or

$$B = \sqrt{\frac{\ln 2}{8\pi^2} \cdot k/n} \cdot \frac{q}{\sigma}. \tag{5}$$

The expression for t is then

$$2^t = \frac{\ln 2}{8\pi^2} \cdot k \cdot q^2/\sigma^2. \tag{6}$$

Each of the t steps should deliver $m = 2^k$ vectors of the form described before.

Since we have two parts in each reduction step, we need to analyze these parts separately. First, consider performing the first part of reduction step number i using coded-BKW with an $[n_i, d_i]$ linear code, where the parameters n_i and d_i at each step are chosen for optimal (global) performance. We sort the 2^k vectors into 2^{d_i} different lists. Here the coded-BKW step guarantees that all the vectors in a list, restricted to the n_i considered positions, have an average norm less than $\sqrt{n_i} \cdot B$ if the codeword is subtracted from the vector. So the number of lists (2^{d_i}) has to be chosen so that this norm restriction is true. Then, after the coded-BKW step, the sieving step should leave the average norm over the N_i positions unchanged, i.e., less than $\sqrt{N_i} \cdot B$.

Since all vectors in a list can be considered to have norm $\sqrt{N_i} \cdot B$ in these N_i positions, the sieving step needs to find any pair that leaves a difference between two vectors of norm at most $\sqrt{N_i} \cdot B$. From the theory of sieving in lattices, we know that heuristics imply that a single list should contain at least $2^{0.208N_i}$ vectors to be able to produce the same number of vectors. The time and space complexity is $2^{0.292N_i}$ if LSF is employed.

Let us adopt some further notation. As we expect the number of vectors to be exponential we write $k = c_0 n$ for some c_0. Also, we adopt $q = n^{c_q}$ and $\sigma = n^{c_s}$. Then $B = C \cdot n^{c_q - c_s}$ and $t = \log_2 D + (2(c_q - c_s) + 1)\log_2 n$, for some constants C and D.

6.1 Asymptotics of Coded-BKW with Sieving

We assume exponential overall complexity and write it as 2^{cn} for some coefficient c to be determined. Each step is additive with respect to complexity, so we assume that we can use 2^{cn} operations in each step.

In the t steps we are choosing $n_1, n_2, \ldots$ positions for each step.

The number of buckets needed for the first step of coded-BKW is $(C' \cdot n^{c_s})^{n_1}$, where C' is another constant. In each bucket the dominant part in the time complexity is the sieving cost $2^{\lambda n_1}$, for a constant λ. The overall complexity, the product of these expressions, should match the bound 2^{cn}, and thus we choose n_1 such that $(C' \cdot n^{c_s})^{n_1} \approx 2^{cn} \cdot 2^{-\lambda n_1}$.

Since $B = C \cdot n^{c_q - c_s}$ we can in the first step use n_1 positions in such a way that $(C' \cdot n^{c_s})^{n_1} \approx 2^{cn} \cdot 2^{-\lambda n_1}$, where C' is another constant and λ is the constant hidden in the exponent of the sieving complexity.

Taking the log, $c_s \log n \cdot n_1 + \log C' n_1 = cn - \lambda n_1$. Therefore, we obtain

$$n_1 = \frac{cn}{c_s \log n + \lambda + \log C'}.$$

To simplify expressions, we use the notation $W = c_s \log n + \lambda + \log C'$.

For the next step, we get $W \cdot n_2 = cn - \lambda n_1$, which simplifies in asymptotic sense to

$$n_2 = \frac{cn}{W}\left(1 - \frac{\lambda}{W}\right).$$

Continuing in this way, we have $W \cdot n_i = cn - \lambda \sum_{j=1}^{i-1} n_j$ and we can obtain an asymptotic expression for n_i as

$$n_i = \frac{cn}{W}\left(1 - \frac{\lambda}{W}\right)^{i-1}.$$

After t steps we have $\sum_{i=1}^{t} n_i = n$, so we observe that

$$\sum_{i=1}^{t} n_i = \frac{cn}{W}\sum_{i=1}^{t}\left(1 - \frac{\lambda}{W}\right)^{i-1},$$

which simplifies to

$$n = \sum_{i=1}^{t} n_i = \frac{cn}{\lambda}\left(1 - \left(1 - \frac{\lambda}{W}\right)^{t}\right).$$

Now, we know that

$$c = \lambda\left(1 - \left(1 - \frac{\lambda}{W}\right)^{t}\right)^{-1}.$$

Since t and W are both of order $\Theta(\log n)$ that tend to infinity as n tends to infinity, we have that

$$c = \lambda\left(1 - \left(1 - \frac{\lambda}{W}\right)^{\frac{W}{\lambda}\cdot\frac{t\lambda}{W}}\right)^{-1} \to \lambda\left(1 - e^{-\frac{t\lambda}{W}}\right)^{-1},$$

when $n \to \infty$.

Since $t/W \to \left(1 + 2\left(c_q - c_s\right)\right)/c_s$ when $n \to \infty$ this finally gives us

$$c = \frac{\lambda}{1 - e^{-\lambda(1+2(c_q-c_s))/c_s}}.$$

Theorem 1. *The time and space complexity of the proposed algorithm is* $2^{(c+o(1))n}$*, where*

$$c = \frac{\lambda}{1 - e^{-\lambda(1+2(c_q-c_s))/c_s}},$$

and $\lambda = 0.292$ *for classic computers and* 0.265 *for quantum computers.*

Proof. Since $c > \lambda$, there are exponential samples left for the distinguishing process. One can adjust the constants in (5) and (6) to ensure the success probability of hypothesis testing close to 1. □

6.2 Asymptotics when Using Plain BKW Pre-Processing

In this section we show that Theorem 1 can be improved for certain LWE parameters. Let $t = t_0 + t_1$. We first derive the following lemma.

Lemma 1. *It is asymptotically beneficial to perform t_0 plain BKW steps, where t_0 is of order $\left(2\left(c_q - c_s\right) + 1 - c_s/\lambda \cdot \ln\left(c_q/c_s\right)\right)\log n$, if*

$$\frac{c_s}{\lambda} \ln \frac{c_q}{c_s} < 2\left(c_q - c_s\right) + 1.$$

Proof. Suppose in each plain BKW steps, we zero-out b positions. Therefore, we have that

$$q^b = \tilde{\mathcal{O}}\left(2^{cn}\right),$$

and it follows that asymptotically b is of order $cn/(c_q \log n)$.

Because the operated positions in each step will decrease using coded-BKW with sieving, it is beneficial to replace a step of coded-BKW with sieving by a pre-processing step of plain BKW, if the allowed number of steps is large. We compute t_1 such that for $i \geq t_1$, $n_i \leq b$. That is,

$$\frac{cn}{W}\left(1 - \frac{\lambda}{W}\right)^{t_1 - 1} = \frac{cn}{c_q \log n}.$$

Thus, we derive that t_1 is of order $c_s/\lambda \cdot \ln\left(c_q/c_s\right) \cdot \log n$. □

If we choose $t_0 = t - t_1$ plain BKW steps, where t_1 is of order $c_s/\lambda \cdot \ln\left(c_q/c_s\right) \cdot \log n$ as in Lemma 1, then

$$n - t_0 b = \sum_{i=1}^{t_1} n_i = \frac{cn}{\lambda}\left(1 - \left(1 - \frac{\lambda}{W}\right)^{t_1}\right).$$

Thus

$$1 - \frac{c}{c_q}\left(2\left(c_q - c_s\right) + 1 - \frac{c_s}{\lambda}\ln\left(\frac{c_q}{c_s}\right)\right) = \frac{c}{\lambda}\left(1 - \frac{c_s}{c_q}\right).$$

Finally, we have the following theorem for characterizing its asymptotic complexity.

Theorem 2. *If $c > \lambda$ and $\frac{c_s}{\lambda}\ln\frac{c_q}{c_s} < 2\left(c_q - c_s\right) + 1$, then the time and space complexity of the proposed algorithm with plain BKW pre-processing is $2^{(c+o(1))n}$, where*

$$c = \left(\lambda^{-1}\left(1 - \frac{c_s}{c_q}\right) + c_q^{-1}\left(2\left(c_q - c_s\right) + 1 - \frac{c_s}{\lambda}\ln\left(\frac{c_q}{c_s}\right)\right)\right)^{-1},$$

and $\lambda = 0.292$ for classic computer and 0.265 for quantum computers.

Proof. The proof is similar to that of Theorem 1. □

Table 1. Asymptotic complexity of Regev's LWE parameter setting

Algorithm	Complexity exponent (c)
QS-BKW(w/ p)	0.8856
S-BKW(w/ p)	0.8951
S-BKW(w/o p)	0.9054
BKW2	0.9299
ENUM/DUAL	4.6720
DUAL-EXPSamples	1.1680

6.3 CaseStudy: Asymptotic Complexity of Regev's Instances

In this part we present a case-study on the asymptotic complexity of Regev's parameter sets, a family of LWE instances with significance in public key cryptography.

Regev's instances: We pick that $q \approx n^2$ and $\sigma = n^{1.5}/(\sqrt{2\pi}\log_2^2 n)$ as suggested in [42].

The asymptotic complexity of Regev's LWE instances is shown in Table 1. For this parameter set, we have $c_q = 2$ and $c_s = 1.5$, and the previously best algorithms in the asymptotic sense are the coded-BKW variants [23, 29] (denoted BKW2 in this table) with time complexity $2^{0.9299n+o(n)}$. The item ENUM/DUAL represents the run time exponent of lattice reduction approaches using polynomial samples and exponential memory, while DUAL-EXPSamples represents the run time exponent of lattice reduction approaches using exponential samples and memory. Both values are computed according to formulas from [25], i.e., $2c_{BKZ} \cdot c_q/(c_q - c_s)^2$ and $2c_{BKZ} \cdot c_q/(c_q - c_s + 1/2)^2$, respectively. Here c_{BKZ} is chosen to be 0.292, the best constant that can be achieved heuristically [9].

We see from the table that the newly proposed algorithm coded-BKW with sieving outperforms the previous best algorithms asymptotically. For instance, the simplest strategy without plain BKW pre-processing, denoted S-BKW(w/o p), costs $2^{0.9054n+o(n)}$ operations, with pre-processing, the time complexity, denoted S-BKW(w/ p) is $2^{0.8951n+o(n)}$. Using quantum computers, the constant hidden in the exponent can be further reduced to 0.8856, shown in the table as QS-BKW(w/ p). Note that the exponent of the algebraic approach is much higher than that of the BKW variants for Regev's LWE instances.

6.4 A Comparison with the Asymptotic Complexity of Other Algorithms

A comparison between the asymptotic time complexity of coded-BKW with sieving and the previous best single-exponent algorithms is shown in Figure Figure 2, similar to the comparison made in [26]. We use pre-processing with

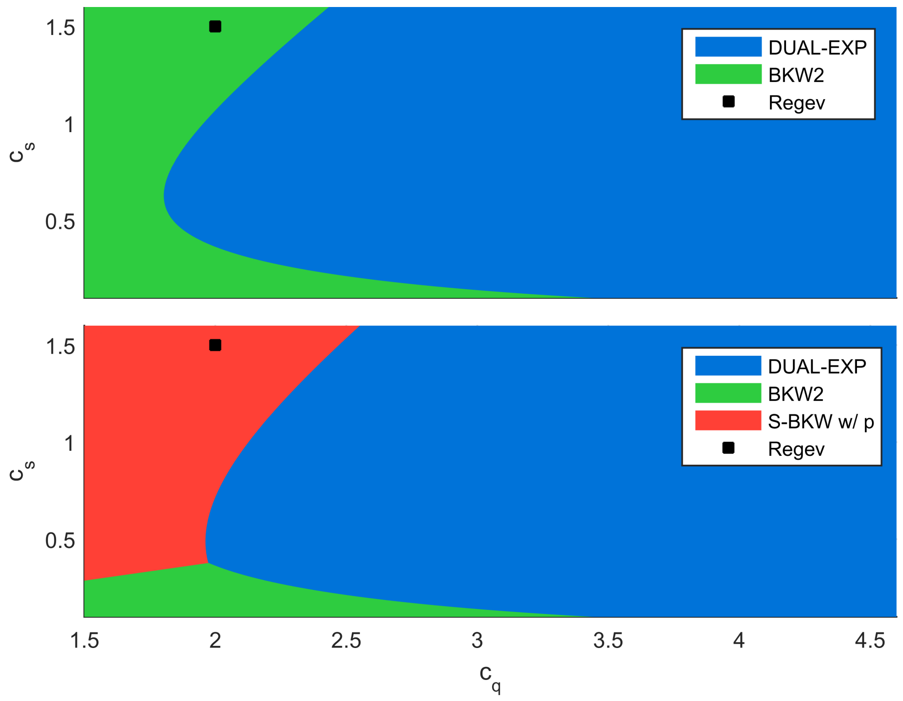

Fig. 2. A comparison between the asymptotic behavior of the best single-exponent algorithms for solving the LWE problem for different values of c_q and c_s. The different colored areas correspond to where the corresponding algorithm beats the other algorithms in that subplot.

standard BKW steps (see theorem 2), since that reduces the complexity of the coded-BKW with sieving algorithm for the entire plotted area.

Use of exponential space is assumed. Access to an exponential number of samples is also assumed. In the full version of this paper we will add a comparison between coded-BKW with sieving and the other algorithms, when only having access to a polynomial number of samples.

First of all we notice that coded-BKW with sieving behaves best for most of the area where BKW2 used to be the best algorithm. It also outperforms the dual algorithm with exponential number of samples on some areas where that algorithm used to be the best. It is also worth mentioning that the Regev instance is well within the area where coded-BKW with sieving performs best.

The area where BKW2 outperforms coded-BKW with sieving is not that cryptographically interesting, since $c_s < 0.5$, and thus Regev's reduction proof does not apply. By also allowing pre-processing with coded-BKW steps our new algorithm is guaranteed to be the best BKW algorithm. We will add an asymptotic expression for this version of the algorithm and update the plot in the full version of our paper.

It should be noted that the Arora-Ge algorithm has the best time complexity of all algorithms for $c_s < 0.5$.

7 Instantiations for Better Concrete Complexity

Until now we have described the new algorithm and analyzed its asymptotic performance. In this section, the practical instantiations for a better concrete solving complexity will be further discussed. We will include techniques and heuristics that may be negligible in the asymptotic sense, but has significant impact in practice.

Algorithm 2 Concrete instantiation of Algorithm 1

```
Input: Matrix $\mathbf{A}$ with $n$ rows and $m$ columns, received vector $\mathbf{z}$ of length $m$
and parameters $t = t_0 + t_{1a} + t_{1b}$, $n_i$ values and $B$.
    change the distribution of the secret vector (Gaussian elimination)
    perform $t_0$ plain BKW steps
    perform $t_{1a}$ coded-BKW steps
    perform $t_{1b}$ steps of coded-BKW with sieving
    perform subspace hypothesis testing using FFT
```

The instantiation of Algorithm 1 for a good concrete complexity is shown in Algorithm 2, consisting of four main steps other than the pre-processing of changing the secret distribution. We include some coded-BKW steps which can be more efficient than sieving when the dimension of the elimination part of the vectors is small. Similar to [23], we also change the final step of guessing and performing hypothesis tests to a step of subspace hypothesis testing using the Fast Fourier Transformation (FFT). The required number of samples can thus be estimated as

$$8 \cdot l \cdot \log_2(q) \cdot e^{4\pi^2 \cdot \frac{\sigma_{final}^2}{q^2}},$$

where l is the dimension of the lattice codes employed in the last subspace hypothesis testing steps, and σ_{final}^2 is the variance of final noise variable.

In the computation, we also assume that the complexity for solving the SVP problem with dimension k using heuristic sieving algorithms is $2^{0.292k+16.4}$ for $k > 90$; otherwise, for a smaller dimension k, the constant hidden in the run time exponent is chosen to be 0.387. This estimation was used by Albrecht et al. in their LWE-estimator [6]. They in turn derived the constant 16.4 from the experimental result in [30].

Table 2. Concrete time complexity for solving Regev's LWE instances.

n	q	σ	Complexity ($\log_2 \#\mathbb{Z}_2$)	
			This paper	[†]Coded-BKW [23]
256	65,537	25.53	134	149
512	262,147	57.06	252	292

[†]: This estimation is also adopted in Albrecht's LWE-estimator [6].

A conservative estimate of the concrete complexity for solving Regev's LWE instances with dimension 256 and 512 is shown in Table 2. The complexity numbers are much smaller than those in [23], and the improvement is more significant when $n = 512$, with a factor of about 2^{40}. Therefore, other than its asymptotic importance, it can be argued that the algorithm coded-BKW with sieving is also efficient in practice.

We compare with the complexity numbers shown in [23] since their estimation also excludes the unnatural selection heuristic. Actually, we could have smaller complexity numbers by taking the influence of this heuristic into consideration, similar to what was done in the eprint version of [29].

The unnatural selection technique is a very useful heuristic to reduce the variance of noise in the vectorial part, as reported in the simulation section of [23]. On the other hand, the analysis of this heuristic is a bit tricky, and should be treated carefully.

We have implemented the new algorithm and tested it on small LWE instances. For ease of implementation, we chose to simulate using the LMS technique together with Laarhoven's angular LSH strategy [30]. While the simulation results bear no implications for the asymptotics, the performance for these small instances was as expected.

8 Conclusions and Future Work

In the paper we have presented a new BKW-type algorithm for solving the LWE problem. This algorithm, named coded-BKW with sieving, combines important ideas from two recent algorithmic improvements in lattice-based cryptography, i.e., coded-BKW and heuristic sieving for SVP, and outperforms the previously known approaches for important parameter sets in public-key cryptography.

For instance, considering Regev's parameter set, we have demonstrated an exponential asymptotic improvement, reducing the time and space complexity from $2^{0.930n}$ to $2^{0.895n}$ in the classic setting. We also obtain the first quantum acceleration for this parameter set, further reducing the complexity to $2^{0.886n}$ if quantum computers are provided.

This algorithm also has significant non-asymptotic improvements for some concrete parameters, compared with the previously best BKW variants. But one should further investigate the analysis when heuristics like unnatural selection are taken into consideration, in order to fully exploit its power on suggesting accurate security parameters for real cryptosystems.

The newly proposed algorithm definitely also has importance in solving many LWE variants with specific structures, e.g., the Binary-LWE problem and the Ring-LWE problem. An interesting research direction is to search for more applications, e.g., solving hard lattice problems, as in [29].

References

1. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the thirty-third annual ACM symposium on Theory of computing. pp. 601–610. ACM (2001)
2. Albrecht, M., Cid, C., Faugere, J.C., Robert, F., Perret, L.: Algebraic algorithms for LWE problems. Cryptology ePrint Archive, Report 2014/1018 (2014)
3. Albrecht, M.R., Cid, C., Faugere, J.C., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. Designs, Codes and Cryptography 74(2), 325–354 (2015)
4. Albrecht, M.R., Faugère, J.C., Fitzpatrick, R., Perret, L.: Lazy Modulus Switching for the BKW Algorithm on LWE. In: Krawczyk, H. (ed.) Public-Key Cryptography–PKC 2014, Lecture Notes in Computer Science, vol. 8383, pp. 429–445. Springer Berlin Heidelberg (2014), *http://dx.doi.org/10.1007/978-3-642-54631-0_25*
5. Albrecht, M.R., Fitzpatrick, R., Göpfert, F.: On the efficacy of solving LWE by reduction to unique-SVP. In: International Conference on Information Security and Cryptology. pp. 293–310. Springer (2013)
6. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology 9(3), 169–203 (2015)
7. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In: Halevi, S. (ed.) Advances in Cryptology–CRYPTO 2009, Lecture Notes in Computer Science, vol. 5677, pp. 595–618. Springer Berlin Heidelberg (2009), *http://dx.doi.org/10.1007/978-3-642-03356-8_35*

8. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Automata, Languages and Programming, pp. 403–415. Springer (2011)
9. Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 10–24. Society for Industrial and Applied Mathematics (2016)
10. Becker, A., Gama, N., Joux, A.: A sieve algorithm based on overlattices. LMS Journal of Computation and Mathematics 17(A), 49–70 (2014)
11. Bernstein, D.J., Lange, T.: Never trust a bunny. In: Radio Frequency Identification. Security and Privacy Issues, pp. 137–148. Springer (2013)
12. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM 50(4), 506–519 (2003)
13. Bogos, S., Vaudenay, S.: Optimization of LPN solving algorithms. In: Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22. pp. 703–728. Springer (2016)
14. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Advances in Cryptology–CRYPTO 2012, pp. 868–886. Springer (2012)
15. Brakerski, Z., Vaikuntanathan, V.: Efficient Fully Homomorphic Encryption from (Standard) LWE. In: Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science. pp. 97–106. IEEE Computer Society (2011)
16. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Annual Cryptology Conference. pp. 505–524. Springer (2011)
17. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Advances in Cryptology–ASIACRYPT 2011, pp. 1–20. Springer (2011)
18. Conway, J. H., Sloane, N. J. A.: Sphere packings, lattices and groups. In: (Vol. 290). Springer Science and Business Media (2013)
19. Dubiner, M.: Bucketing coding and information theory for the statistical high-dimensional nearest-neighbor problem. IEEE Transactions on Information Theory 56(8), 4166–4179 (2010)
20. Duc, A., Tramèr, F., Vaudenay, S.: Better Algorithms for LWE and LWR. In: Advances in Cryptology – EUROCRYPT 2015, pp. 173–202. Springer (2015)
21. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013, pp. 75–92. Springer (2013)
22. Guo, Q., Johansson, T., Löndahl, C.: Solving LPN using covering codes. In: Advances in Cryptology–ASIACRYPT 2014, pp. 1–20. Springer (2014)
23. Guo, Q., Johansson, T., Stankovski, P.: Coded-BKW: Solving LWE using lattice codes. In: Advances in Cryptology–CRYPTO 2015, pp. 23–42. Springer (2015)
24. Hanrot, G., Pujol, X., Stehlé, D.: Algorithms for the shortest and closest lattice vector problems. In: Coding and Cryptology, pp. 159–190. Springer (2011)
25. Herold, G., Kirshanova, E., May, A.: On the asymptotic complexity of solving LWE. IACR Cryptology ePrint Archive 2015, 1222 (2015), *http://eprint.iacr.org/2015/1222*
26. Herold, G., Kirshanova, E., May, A.: On the asymptotic complexity of solving LWE. J. Designs, Codes and Cryptography, pp. 1–29 (2017)
27. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing. pp. 604–613. ACM (1998)

28. Kirchner, P.: Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377 (2011), *http://eprint.iacr.org/*
29. Kirchner, P., Fouque, P.A.: An improved BKW algorithm for LWE with applications to cryptography and lattices. In: Advances in Cryptology–CRYPTO 2015, pp. 43–62. Springer (2015)
30. Laarhoven, T.: Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In: Annual Cryptology Conference. pp. 3–22. Springer (2015)
31. Laarhoven, T., Mosca, M., Van De Pol, J.: Finding shortest lattice vectors faster using quantum search. Designs, Codes and Cryptography 77(2-3), 375–400 (2015)
32. Laarhoven, T., de Weger, B.: Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing. In: International Conference on Cryptology and Information Security in Latin America. pp. 101–118. Springer (2015)
33. Levieil, É., Fouque, P.A.: An improved LPN algorithm. In: Prisco, R.D., Yung, M. (eds.) SCN. Lecture Notes in Computer Science, vol. 4116, pp. 348–359. Springer-Verlag (2006)
34. Lindner, R., Peikert, C.: Better Key Sizes (and Attacks) for LWE-Based Encryption. In: Kiayias, A. (ed.) Topics in Cryptology–CT-RSA 2011, Lecture Notes in Computer Science, vol. 6558, pp. 319–339. Springer Berlin Heidelberg (2011), *http://dx.doi.org/10.1007/978-3-642-19074-2_21*
35. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: Topics in Cryptology–CT-RSA 2013, pp. 293–309. Springer (2013)
36. May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. pp. 203–228 (2015)
37. Micciancio, D., Regev, O.: Lattice-based Cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer Berlin Heidelberg (2009)
38. Micciancio, D., Voulgaris, P.: Faster exponential time algorithms for the shortest vector problem. In: Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms. pp. 1468–1480. SIAM (2010)
39. Mulder, E.D., Hutter, M., Marson, M.E., Pearson, P.: Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA: extended version. J. Cryptographic Engineering 4(1), 33–45 (2014), *http://dx.doi.org/10.1007/s13389-014-0072-z*
40. Nguyen, P.Q., Vidick, T.: Sieve algorithms for the shortest vector problem are practical. J. Mathematical Cryptology 2(2), 181–207 (2008), *http://dx.doi.org/10.1515/JMC.2008.009*
41. Pujol, X., Stehlé, D.: Solving the shortest lattice vector problem in time $2^{2.465n}$. IACR Cryptology ePrint Archive 2009, 605 (2009), *http://eprint.iacr.org/2009/605*
42. Regev, O.: On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. Journal of the ACM 56(6), 34:1–34:40 (Sep 2009), *http://doi.acm.org/10.1145/1568318.1568324*
43. Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical programming 66(1-3), 181–199 (1994)
44. Wagner, D.: A generalized birthday problem. In: Advances in cryptology–CRYPTO 2002, pp. 288–304. Springer (2002)

45. Wang, X., Liu, M., Tian, C., Bi, J.: Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. pp. 1–9. ACM (2011)
46. Zhang, B., Jiao, L., Wang, M.: Faster algorithms for solving LPN. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 168–195. Springer (2016)
47. Zhang, F., Pan, Y., Hu, G.: A three-level sieve algorithm for the shortest vector problem. In: International Conference on Selected Areas in Cryptography. pp. 29–47. Springer (2013)