Coded-BKW: Solving LWE Using Lattice Codes

Qian Guo[1,2], Thomas Johansson[1(✉)], and Paul Stankovski[1]

[1] Department of Electrical and Information Technology, Lund University, Lund, Sweden
{qian.guo,thomas.johansson,paul.stankovski}@eit.lth.se
[2] Shanghai Key Laboratory of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai, China

Abstract. In this paper we propose a new algorithm for solving the Learning With Errors (LWE) problem based on the steps of the famous Blum-Kalai-Wasserman (BKW) algorithm. The new idea is to introduce an additional procedure of mapping subvectors into codewords of a lattice code, thereby increasing the amount of positions that can be cancelled in each BKW step. The procedure introduces an additional noise term, but it is shown that by using a sequence of lattice codes with different rates the noise can be kept small. Developed theory shows that the new approach compares favorably to previous methods. It performs particularly well for the BINARY-LWE case, i.e., when the secret vector is sampled from $\{0,1\}^*$.

Keywords: LWE · BINARY-LWE · BKW · Coded-BKW · Lattice codes

1 Introduction

Learning with Errors (LWE) is a problem that has received a lot of attention recently and can be considered as a generalization of the Learning Parity with Noise (LPN) problem. Regev introduced LWE in [31], and it has proved to be a very useful tool for constructing cryptographic primitives. Although a great number of different constructions of cryptographic primitives have been given since the introduction of the LWE problem, one of the most interesting ones is the work on constructing fully homomorphic encryption schemes [8,10,19,20].

There are several motivating reasons for the interest in LWE-based cryptography. One is the simplicity of the constructions, sometimes giving rise to very efficient implementations which run much faster than competing alternative solutions. Another reason is the well-developed theory on lattice problems, which gives insights into the hardness of the LWE problem. There are theoretical reductions from worst-case lattice problems to average-case LWE [31].

Q. Guo—Supported in part by Shanghai Key Program of Basic Research (No. 12JC1401400), the National Defense Basic Research Project (No. JCYJ-1408).
T. Johansson and P. Stankovski—Supported by the Swedish Research Council (Grants No. 621-2012-4259).

R. Gennaro and M. Robshaw (Eds.): CRYPTO 2015, Part I, LNCS 9215, pp. 23–42, 2015.
DOI: 10.1007/978-3-662-47989-6_2

A third motivating reason is the fact that LWE-based cryptography is one of the areas where a quantum computer is not known to be able to break the primitives (contrary to factoring-based and discrete log-based primitives). This is sometimes referred to as being a tool in *post-quantum cryptography.*

Let us state the LWE problem.

Definition 1. *Let n be a positive integer, q an odd 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*

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

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 so-called *decision* LWE problem. In this case the problem is to distinguish samples drawn from $L_{\mathbf{s},\mathcal{X}}$ and samples drawn from a uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$. Typically, we are then interested in distinguishers with non-negligible advantage.

The parameters of an LWE instance are typically chosen with some internal relations. The prime q is chosen as a polynomial in n, and the discrete Gaussian distribution $\mathcal{X}$ has mean zero and standard deviation $\sigma = \alpha \cdot q$ for some small α. For example, in [31], Regev proposed to use parameters $q \approx n^2$ and $\alpha = 1/(\sqrt{2\pi n} \cdot \log_2^2 n)$.

1.1 Previous Work

A number of algorithms for solving the LWE problem have been given, using different approaches. As there is a strong connection to lattice problems, a direction for a subset of the algorithms has been to either rewrite the LWE problem as the problem of finding a short vector in a dual lattice, the Short Integer Solution (SIS) problem, or to solve the Bounded Distance Decoding (BDD) problem. Lattice reduction algorithms may be applied to solve these problems. Even though there has been a lot of research devoted to the study of lattice reduction algorithms, there still seems to be quite some uncertainty about the complexity and performance of such algorithms for higher dimensions.

Another very interesting approach was given by Arora and Ge in [5], where they proposed a novel algebraic approach to solve the LWE problem. The asymptotic complexity of this algorithm is subexponential when $\sigma \leq \sqrt{n}$, but fully exponential otherwise. The algorithm is mainly of asymptotic interest as applying it on specific instances gives higher complexity than other solvers.

Finally, much work has been done on combinatorial algorithms for solving LWE, all taking the famous Blum-Kalai-Wasserman (BKW) algorithm [7] as a basis. The BKW algorithm resembles the generalized birthday approach by

Wagner [34] and was originally given as an algorithm for solving the LPN problem. These combinatorial algorithms have the advantage that their complexity can be analyzed in a standard way and we can get explicit values on the complexity for different instantiations of the LWE problem. Even though we use approximations in the analysis, the deviation between theoretical analysis and actual performance seems to be small [3,17]. This approach tends to give algorithms with the best performance for some important parameter choices. A possible drawback with BKW-based algorithms is that they usually require a huge amount of memory, often of the same order as the time complexity. Some recent work in this direction is [1,3,17].

1.2 Motivation and Contributions

We know that the theoretical hardness of the LWE problem is well-established, through reductions to hard lattice problems [9,30,31]. This can be transferred to asymptotic statements on the security. In fact, most proposals of LWE-based cryptographic primitives rely only on asymptotics when arguing about security.

But there is also a huge interest in studying the actual hardness of specific instances of the LWE problem. How does the choice of parameters (n, q, σ) influence the complexity of solving LWE? What are the smallest parameters we can have and still achieve, say, 80-bit security?

In this paper we introduce a new algorithm for the LWE problem which again uses the BKW algorithm as a basis. The novel idea is to introduce a modified BKW step, we call it a coded-BKW step, that allows us to cancel out more positions in the $\mathbf{a}$ vectors than a traditional BKW step. The coded-BKW step involves mapping the considered part of an $\mathbf{a}$ vector into a nearest codeword in a lattice code (a linear code over $\mathbb{Z}_q$, where the distance is the Euclidean norm). The mapping to the nearest codeword introduces some noise, but with

Table 1. Time complexity comparison for solving various LWE instances.

n	q	σ	Complexity ($\log_2 \#\mathbb{Z}_q$)			
			This paper (Sect. 5)	Duc et al. [17]	NTL-BKZ Lindner-Peikert model [1,25]	BKZ 2.0 Simulator Model [1,11,26]
Regev [31]						
128	16,411	11.81	84.5	95.0	61.6	61.9
256	65,537	25.53	145.1	178.7	175.5	174.5
512	262,147	57.06	287.6	357.5	386.8	518.6
Lindner & Peikert [25]						
128	2,053	2.70	69.7	83.7	54.5	57.1
256	4,099	3.34	123.8	154.2	156.2	151.2
512	4,099	2.90	209.2	271.8	341.9	424.5

Table 2. Time complexity comparison for solving various BINARY-LWE instances.

n	q	σ	Complexity ($\log_2 \#\mathbb{Z}_2$)				
			This paper (Sect. 7)	Albrecht et al. [3]		NTL-BKZ L-P model [1,25]	BKZ 2.0 Sim. model [1,11,26]
				w/o Unnatural Selection	Improved version		
Regev [31]							
128	16,411	11.81	58.8	78.2	74.2	65.4	65.7
256	65,537	25.53	97.9	142.7	132.5	179.5	178.5
512	262,147	57.06	163.7	251.2	241.8	390.9	522.8

a proper selection of parameters this can be kept bounded and small enough not to influence the total noise in the BKW procedure too much. Whenever any pair of $\mathbf{a}$ vectors map to the same codeword, they are added together creating a new sample with a part of the $\mathbf{a}$ vector cancelled, as is the usual result of a BKW step. These samples are the input to the next step in the BKW procedure. The algorithm also contains some additional new steps using the discrete Fast Fourier Transform (FFT) to provide some additional improvement. The new ideas have some connection to the recent paper on LPN [22], but in that paper a binary covering code (see [12] for definition) was used after the BKW steps. Also the recent work [3] has some connections as it introduces additional noise in the BKW steps by the so-called lazy modulus switching. Still, the new algorithm outperforms previous BKW-type algorithms for solving LWE — even when compared with the most recent work [17], we improve significantly (as detailed in Table 1).

We also apply the algorithm in a slightly modified form on the BINARY-LWE problem. The BINARY-LWE problem is the LWE problem when the secret vector $\mathbf{s}$ is chosen uniformly from $\{0,1\}^n$. In this case we have a huge improvement (see Table 2) in performance compared with other algorithms.

Tables 1 and 2 show comparisons of different algorithms for solving various LWE and BINARY-LWE instances, respectively. We compare the performance of the new algorithm with the previous best BKW variant (i.e., Duc et al. [17] for LWE or Albrecht et al. [3] for BINARY-LWE) and the estimates (under certain models [11,25,26,29]) for distinguishing LWE (or BINARY-LWE) samples from uniform using lattice reduction algorithms, when LWE is reduced to SIS. The results consolidate the understanding that BKW is asymptotically efficient. For the toy LWE instances with $n = 128$, the SIS approach still beats all the BKW variants, including ours; but the recent variant has greatly narrowed the gap. The situation alters when the parameter n increases.

We also obtain a significant improvement (i.e., with a factor of more than 2^{11} in time) on solving an LWE $(136, 2003, 5.19)$-instance, which first appeared

in [29] and was then adopted as an example in [25], compared with the estimates in [1] that use the BDD approach.

Thus, we are close to a conclusion that, when choosing LWE instances for today's cryptosystems (e.g., achieving an 80-bit or higher security level), thwarting of BKW-type attacks must be taken into consideration.

The remainder of the paper is organized as follows. In Sect. 2 we describe the basic theory around the LWE problem. We give a short description of the BKW algorithm in Sect. 3, and then present the novel modification in the next section. We detail the algorithm in Sect. 5, analyze its complexity in Sect. 6, and then propose a variant for BINARY-LWE in Sect. 7. This is followed by the sections of implementation and results. We finally concludes this paper in Sect. 10.

2 Background

On an n-dimensional Euclidean space $\mathbb{R}^n$, the intuitive notion of length of a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is captured by the L_2-norm; $||\mathbf{x}|| = \sqrt{x_1^2 + \cdots + x_n^2}$. The Euclidean distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$ is defined as $||\mathbf{x}-\mathbf{y}||$. For a given set of vectors $\mathcal{L}$, the minimum mean square error (MMSE) estimator assigns each vector in $\mathbb{R}^n$ to the vector $\mathbf{l} \in \mathcal{L}$ such that $||\mathbf{x} - \mathbf{l}||$ is minimized. Let us shortly introduce the discrete Gaussian distribution.

2.1 Discrete Gaussian Distribution

Let $x \in \mathbb{Z}$. The discrete Gaussian distribution on $\mathbb{Z}$ with mean 0 and variance σ^2, denoted $D_{\mathbb{Z},\sigma}$, is the probability distribution obtained by assigning a probability proportional to $\exp(-x^2/2\sigma^2)$ to each $x \in \mathbb{Z}$. The $\mathcal{X}$ distribution[1] with variance σ^2 is the distribution on $\mathbb{Z}_q$ obtained by folding $D_{\mathbb{Z},\sigma}$ mod q, i.e., accumulating the value of the probability mass function over all integers in each residue class $mod q$. Similarly, we define the discrete Gaussian over $\mathbb{Z}^n$ with variance σ^2, denoted $D_{\mathbb{Z}^n,\sigma}$, as the product distribution of n independent copies of $D_{\mathbb{Z},\sigma}$.

In general, the discrete Gaussian distribution does not exactly inherit the usual properties from the continuous case, but in our considered cases it will be close enough and we will use properties from the continuous case, as they are approximately correct. For example, if X is drawn from $\mathcal{X}_{\sigma_1}$ and Y is drawn from $\mathcal{X}_{\sigma_2}$, then we consider $X + Y$ to be drawn from $\mathcal{X}_{\sqrt{\sigma_1^2+\sigma_2^2}}$. This follows the path of previous work [1].

A central point in cryptanalysis is to estimate the number of samples required to distinguish between two distributions, in our case the uniform distribution on $\mathbb{Z}_q$ and $\mathcal{X}_\sigma$. The solution to this distinguishing problem leads to an efficient key recovery: we assume that for a right guess, the observed symbol is $\mathcal{X}_\sigma$ distributed; otherwise, it is uniformly random. Thus, we need to distinguish the secret from Q candidates. We follow the theory from linear cryptanalysis [6] (also similar to that in correlation attacks [15]), that the number M of required samples to test

[1] It is also denoted $\mathcal{X}_\sigma$, and we omit σ if there is no ambiguity.

is about $\mathcal{O}\left(\frac{\ln(Q)}{\Delta(\mathcal{X}_\sigma\|U)}\right)$, where $\Delta(\mathcal{X}_\sigma\|U)$ is the divergence[2] between $\mathcal{X}_\sigma$ and the uniform distribution U in $\mathbb{Z}_q$.

2.2 LWE Problem Description

We already gave the definition of the search LWE problem in Definition 1. As we will focus on this problem, we skip giving a more formal definition of the decision version of LWE. Instead we reformulate the search LWE problem a bit. Assume that we ask for m samples from the LWE distribution $L_{\mathbf{s},\mathcal{X}}$ and the response is denoted as

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

where $\mathbf{a}_i \in \mathbb{Z}_q^n, z_i \in \mathbb{Z}_q$. We introduce $\mathbf{z} = (z_1, z_2, \ldots, z_m)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_m) = \mathbf{sA}$. We can then write $\mathbf{A} = \left[\mathbf{a}_1^{\mathrm{T}}\, \mathbf{a}_2^{\mathrm{T}} \cdots \mathbf{a}_n^{\mathrm{T}}\right]$ and $\mathbf{z} = \mathbf{sA} + \mathbf{e}$, where $z_i = y_i + e_i = \langle \mathbf{s}, \mathbf{a}_i \rangle + e_i$ and $e_i \xleftarrow{\$} \mathcal{X}$ is the noise. We see that the problem has been reformulated as a decoding problem. The matrix $\mathbf{A}$ serves as the generator matrix for a linear code over $\mathbb{Z}_q$ and $\mathbf{z}$ is the received word. Finding the codeword $\mathbf{y} = \mathbf{sA}$ such that the distance $||\mathbf{y} - \mathbf{z}||$ is minimum will give the secret vector $\mathbf{s}$.

If the secret vector $\mathbf{s}$ is drawn from the uniform distribution, there is a simple transformation [4,23] that can be applied, namely, we may through Gaussian elimination transform $\mathbf{A}$ into systematic form. Assume that the first n columns are linearly independent and form the matrix $\mathbf{A_0}$. Define $\mathbf{D} = {\mathbf{A_0}}^{-1}$. With a change of variables $\hat{\mathbf{s}} = \mathbf{sD}^{-1} - (z_1, z_2, \ldots, z_n)$ we get 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

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

After this initial step, each entry in the secret vector $\hat{\mathbf{s}}$ is now distributed according to $\mathcal{X}$.

2.3 Lattice Codes and Construction A

A lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$. Reformulated, Λ is a lattice iff there are linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^n$, such that any $\mathbf{y} \in \Lambda$ can be written as $\mathbf{y} = \sum_{i=1}^{m} \alpha_i \mathbf{v}_i$, where $\alpha_i \in \mathbb{Z}$. The set $\mathbf{v}_1, \ldots, \mathbf{v}_m$ is called a basis for Λ. A matrix whose columns are these vectors is said to be a generator matrix for Λ.

Furthermore, let $Vol(\cdot)$ denote the volume of a closed set in $\mathbb{R}^n$ and let $\mathcal{V}$ be the fundamental Voronoi region of Λ, i.e.,

$$\mathcal{V} = \{\mathbf{x} \in \mathbb{R}^n : ||\mathbf{x}|| \leq ||\mathbf{x} - \mathbf{w}||, \forall \mathbf{w} \in \Lambda\}.$$

[2] Divergence has a couple of aliases in literature: relative entropy, information divergence, Kullback-Leibler divergence, etc. We refer the interested reader to [15] for the rigorous definition. In this paper, the divergence $\Delta(\mathcal{X}_\sigma\|U)$ will be computed numerically.

We will be interested in a more narrow class of lattices based on q-ary linear codes. If $\mathcal{C}$ is a linear $[N, k]$ code over the alphabet of size q, where q is a prime, then a lattice over this code is

$$\Lambda(\mathcal{C}) = \{\lambda \in \mathbb{R}^n : \lambda = (\mathbf{c} \bmod q), \mathbf{c} \in \mathcal{C}\}.$$

The mapping from the code to the lattice is often referred to as Construction A [14]. The lattice $\Lambda(\mathcal{C})$ is called the q-ary lattice associated with $\mathcal{C}$.

A typical application is to use the lattice $\Lambda(\mathcal{C})$ as a codebook for quantization of sequences $\mathbf{x} \in \mathbb{R}^n$. Let $Q(\mathbf{x})$ be the lattice point closest to $\mathbf{x}$ if the squared error is used as a fidelity criterion. We call $Q(\mathbf{x})$ an MSE quantizer. The second moment of Λ, denoted by $\sigma^2 = \sigma^2(\Lambda)$, is defined as the second moment per dimension of a uniform distribution over its fundamental region $\mathcal{V}$, i.e.,

$$\sigma^2 = \frac{1}{n} \cdot \int_{\mathcal{V}} ||\mathbf{x}||^2 \frac{1}{Vol(\mathcal{V})} d\mathbf{x}. \tag{1}$$

From the literature [13,14] of lattice codes, we know that the value σ^2 can be represented as

$$\sigma^2 = G(\Lambda) \cdot Vol(\mathcal{V})^{\frac{2}{n}}, \tag{2}$$

where $G(\Lambda)$ is called the normalized second moment, which represents a figure of merit of a lattice quantizer with respect to the MSE distortion measure. We denote the minimum possible value of $G(\Lambda)$ over all lattices in $\mathbb{R}^n$ by $G(\Lambda_n)$ and it is known that

$$\frac{1}{2\pi e} < G(\Lambda_n) \leq \frac{1}{12}, \tag{3}$$

where the upper bound is achieved when the lattice is generated by $\mathbb{Z}^n$, and the lower one is achieved asymptotically by lattices generated by Construction A from q-ary random linear codes [18,27,35].

3 The BKW Algorithm

The BKW algorithm was proposed by Blum et al. [7] and was originally targeting the LPN problem. However, it is trivially adopted also to the LWE problem.

As with Wagner's generalized birthday algorithm, the BKW approach uses an iterative collision procedure on the columns in the generator matrix $\mathbf{A}$, which step by step reduces the dimension of $\mathbf{A}$. Summing together columns that collide in some subset of positions and keeping them as columns in a new matrix reduces the dimension but increases the size of the noise.

A brief description inspired by the notation in [22] follows. Initially, one searches for all combinations of two columns in $\mathbf{A}$ with the same last b entries. Assume that one finds two columns $\mathbf{a}_{i_1}^{\mathrm{T}}, \mathbf{a}_{i_2}^{\mathrm{T}}$ such that

$$\mathbf{a}_{i_1} - \mathbf{a}_{i_2} = (* \; * \; \cdots \; * \underbrace{0 \, 0 \, \cdots \, 0}_{b \text{ symbols}}),$$

where $*$ means any value. Then a new vector $\mathbf{a}_1^{(2)} = \mathbf{a}_{i_1} - \mathbf{a}_{i_2}$ is formed. An "observed symbol" is also formed, corresponding to this new column by forming $z_1^{(2)} = z_{i_1} - z_{i_2}$. If $y_1^{(2)} = \left\langle \mathbf{s}, \mathbf{a}_1^{(2)} \right\rangle$, then $z_1^{(2)} = y_1^{(2)} + e_1^{(2)}$, where now $e_1^{(2)} = e_{i_1} - e_{i_2}$. Recall that noise like e_{i_1} follows the Gaussian distribution with variance σ^2, so $e_1^{(2)} = e_{i_1} - e_{i_2}$ is considered to be Gaussian distributed with variance $2\sigma^2$. There is also a second obvious way of getting collisions, namely, combining any two vectors where the sum of the collision sets is zero. The procedure is analog to the above, just replacing subtraction with addition.

There are different approaches to realizing the above merging procedure. We consider the approach called LF1 in [24], which computes the difference between one fixed column and any other column with the same last b entries (in absolute value), and forwards this to the next BKW step.

Put all such new columns in a matrix $\mathbf{A}_2$,

$$\mathbf{A}_2 = (\mathbf{a}_1^{(2)\mathrm{T}} \mathbf{a}_2^{(2)\mathrm{T}} \dots \mathbf{a}_{m_2}^{(2)\mathrm{T}}).$$

If m is the number of columns in $\mathbf{A}$, then we have the number of columns in $\mathbf{A}_2$ to be $m_2 = m - \frac{q^b-1}{2}$. Hence, using the LF1 approach, the number of samples (columns) forwarded to the next step of BKW is slowly decreasing (by $\frac{q^b-1}{2}$ for each step). It is known from simulation, that the LF2 approach [24] which gives more surviving samples, performs well and could be chosen in an actual implementation.

Now the last b entries of columns in $\mathbf{A}_2$ are all zero. In connection to this matrix, the vector of observed symbols is

$$\mathbf{z}_2 = (z_1^{(2)} z_2^{(2)} \cdots z_{m-\frac{q^b-1}{2}}^{(2)}),$$

where $z_i^{(2)} - y_i^{(2)}$ are assumed Gaussian with variance $2\sigma^2$, for $1 \le i \le m - \frac{q^b-1}{2}$. This completes one step of the BKW algorithm.

We then iterate the same for $i = 2, 3, \dots, t$, picking a new collision set of size $\frac{q^b-1}{2}$ and finding colliding columns in $\mathbf{A}_i$, giving new vectors with an additional b entries being zero, forming the columns of $\mathbf{A}_{i+1}$. Repeating the same procedure an additional $t-2$ times will reduce the number of unknowns in the secret vector $\mathbf{s}$ to $n - bt$ in the remaining problem.

For each iteration the noise is increased. After t BKW steps the noise connected to each column is of the form

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

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

Altogether we have reduced the LWE instance to a smaller instance, where now the length of the secret vector is $n' = n - tb$, but the noise has variance $2^t \cdot \sigma^2$. The remaining unknown part of the secret vector $\mathbf{s}$ is guessed (a total

of q^{n-tb}) and for each guess we check through a hypothesis test whether the remaining samples follow the Gaussian distribution. The number of remaining samples is at least $m - t \cdot \frac{q^b-1}{2}$.

Note that there is an improved version of BKW using lazy modulus reduction [3] and the very recent improvement in [17].

4 A Modified BKW Algorithm for the LWE Problem

The new algorithm we propose uses the same structure as the BKW algorithm. The new idea involves changing the BKW step to a more advanced step that can remove more positions in the treated vectors at the expense of leaving an additional noise term.

We introduce some additional notation. For the index set I, we make use of $\mathbf{v}_I$ to denote the vector with entries indexed by I. Alternatively, we utilize the symbol $\mathbf{v}_{[1,\ldots,n]}$ to denote the vector containing the first n entries of $\mathbf{v}$, etc.

4.1 A New BKW Step

Recall the BKW step, taking a large number of vectors $\mathbf{a}_i$ and trying to collide them in a set of positions determined by an index set I. This part of the vector $\mathbf{a}$ is written as $\mathbf{a}_I$. The size of the collision set ($\frac{q^b-1}{2}$) and the number of vectors have to be of the same order, which essentially determines the complexity of the BKW algorithm, as the number of steps we can perform is determined by the variance of the noise.

We propose to do the BKW step in a different manner. Assuming that we are considering step i in the BKW process, we fix a q-ary linear code with parameters (N_i, b), called $\mathcal{C}_i$. The code gives rise to a lattice code. Now, for any given vector $\mathbf{a}_I$ as input to this BKW step, we approximate the vector by one of the codewords in the code $\mathcal{C}_i$.

We rewrite $\mathbf{a}_I$ into two parts, the codeword part $\mathbf{c}_I \in \mathcal{C}_i$ and an error part $\mathbf{e}_I \in \mathbb{Z}_q^{N_i}$, i.e.,

$$\mathbf{a}_I = \mathbf{c}_I + \mathbf{e}_I. \tag{4}$$

Clearly, we desire the error part to be as small as possible, so we adopt a decoding procedure to find the nearest codeword in the chosen code $\mathcal{C}_i$ using the Euclidean metric. Here, we utilize syndrome decoding by maintaining a large syndrome table, and details will be discussed thoroughly later.

Each vector $\mathbf{a}_I$ is then sorted according to which codeword it was mapped to. Altogether, there are q^b possible codewords. Finally, generate new vectors for the next BKW step by subtracting vectors mapped to the same codeword (or adding to the zero codeword).

The inner product $\langle \mathbf{s}_I, \mathbf{a}_I \rangle$ is equal to

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

By subtracting two vectors mapped to the same codeword we cancel out the first part of the right hand side and we are left with the noise. The latter term is referred to as the error term introduced by coding.

Let us examine the samples we have received after t BKW steps of this kind. In step i we have removed N_i positions, so in total we have now removed $\sum_{i=1}^{t} N_i$ positions ($N_i \geq b$). The received samples are created from summing 2^t original samples, so after guessing the remaining symbols in the secret vector and adjusting for its contribution, a received symbol z can be written as a sum of noise variables,

$$z = \sum_{j=1}^{2^t} e_{i_j} + \sum_{i=1}^{n} s_i(E_i^{(1)} + E_i^{(2)} + \cdots + E_i^{(t)}), \tag{5}$$

where $E_i^{(h)} = \sum_{j=1}^{2^{t-h+1}} \hat{e}_{i_j}^{(h)}$ and $\hat{e}_{i_j}^{(h)}$ is the coding noise introduced in step h of the modified BKW algorithm. Note that on one position i, at most one error term $E_i^{(h)}$ is non-zero.

We observe that noise introduced in early steps is increased exponentially in the remaining steps, so the procedure will use a sequence of codes with decreasing rate. In this way the error introduced in early steps will be small and then it will eventually increase.

4.2 Analyzing the Error Distribution

There are many approaches to estimating the error distribution introduced by coding. The simplest way is just assuming that the value is a summation of several independent discrete Gaussian random variables. This estimation is easily performed and fairly accurate. A second approach is to compute the error distribution accurately (to sufficient precision) by computer. We should note that the error distribution is determined from the linear code employed. We now rely on some known result on lattice codes to provide a good estimate on the size of the noise introduced by coding.

We assume that the error vector $\mathbf{e}$ introduced by the coding technique remains discrete Gaussian, and their summation is discrete Gaussian as well, just as in previous research. As the error is distributed symmetrically we should estimate the value $\mathrm{E}[||\mathbf{e}||^2]$ to bound the effect of the error, where $\mathbf{e}$ is the error vector distributed uniformly on the integer points inside the fundamental region $\mathcal{V}$ of the lattice generated by Construction A.

Thus, the problem of decoding transforms to an MMSE quantizing problem over the corresponding lattice. For simplicity of analysis, we change the hypothesis and assume that the error vector $\mathbf{e}$ is distributed uniformly and continuously on $\mathcal{V}$. Thus we can utilize the theory on lattice codes to give a fairly accurate estimation of the value $\frac{1}{N}\mathrm{E}[||\mathbf{e}||^2]$, which exactly corresponds to the second moment of the lattice σ^2. As given in Eq. (2), we can write it as,

$$\sigma^2 = G(\Lambda) \cdot Vol(\mathcal{V})^{\frac{2}{N}}.$$

In our scheme, although we employ several different linear codes with different rates, we also try to make the contribution of every dimension equal. We generate a lattice Λ by Construction A, given a linear code. We denote the minimum

possible value of $G(\Lambda)$ over all lattices Λ in $\mathbb{Z}^n$ generated by Construction A from an $[N, k]$ linear code as $G(\Lambda_{N,k})$.

Definitely $G(\Lambda_{N,k})$ is no less than $G(\Lambda_N)$; thus it is lower bounded by the value $\frac{1}{2\pi e}$ and this bound can be achieved asymptotically. For the lattice generated by $\mathbb{Z}^N$, i.e., employing a trivial linear code without redundancy, its normalized second moment is $\frac{1}{12}$. Therefore, the value $G(\Lambda_{N,k})$ satisfies

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

We set $G(\Lambda_{N,k})$ to be $\frac{1}{12}$ and surely this is a pessimistic estimation. Since the lattice is built from a linear code by Construction A, the volume of $\mathcal{V}$ is q^{N-k}. Thus, we can approximate σ by

$$\sigma \approx q^{1-k/N} \cdot \sqrt{G(\Lambda_{N,k})} = \frac{q^{1-k/N}}{\sqrt{12}}. \tag{6}$$

We have numerically tested the smallest possible variance of errors introduced by coding, given several small sizes of N, k and q, (e.g., $[N, k]$ is $[3, 1]$ or $[2, 1]$, q is 631, 2053 or 16411) and verified that the above estimation works (see Table 3, where $1/G$ is bounding $1/G(\Lambda_{N,k})$). We choose $[N, 1]$ codes since for the covering or MMSE property, lower rate means worse performance.

It is folklore that the value G will decrease when the dimension and length becomes larger, and all the cases listed in Table 3 fully obey the rule. Thus we believe that we may have even better performance when employing a more complicated code for a larger problem. Actually, the values without a † sign in Table 3 is computed using randomly chosen linear codes, and they still outperform our estimation greatly. This observation fits the theory well that when the dimension n is large, a random linear code may act nearly optimally.

From Eq. (6) we know the variance of the error term from the coding part. Combining this with Eq. (5), we get an estimation of the variance of the total noise for the samples that we create after t modified BKW steps.

4.3 Decoding Method and Constraint

Here we discuss details of syndrome decoding and show that the additional cost is under control. Generally, we characterize the employed $[N, k]$ linear code by a systematic generator matrix $\mathbf{M} = \left[\mathbf{I}\,\mathbf{F}'\right]_{k\times N}$. Thus, a corresponding parity-check matrix $\mathbf{H} = \left[\mathbf{F}'^{\mathrm{T}}\,\mathbf{I}\right]_{(\mathbf{N}-\mathbf{k})\times\mathbf{N}}$ is directly obtained.

Table 3. Numerical evaluations on $1/G$

q	631			2053			16411	
code	[2,1]	[3,1]	[4,1]	[2,1]	[3,1]	[4,1]	[2,1]	[3,1]
$\mathrm{E}[\|\|\mathbf{e}\|\|^2]$	$101.26^{\dagger}$	1277.31	4951.53	$329.24^{\dagger}$	6185.67	29107.73	$2631.99^{\dagger}$	99166.25
$1/G$	12.46	12.71	12.80	12.47	12.65	12.78	12.47	12.62

The value with a † sign means that it is optimal.

The syndrome decoding procedure is described as follows. (1) We construct a constant-time query table containing q^{N-k} items, in each of which we store the syndrome and its corresponding error vector with minimum Euclidean distance. (2) When the syndrome is computed, by checking the table, we locate its corresponding error vector and add them together, thereby yielding the desired nearest codeword.

We generalize the method in [22] to the non-binary case $\mathbb{Z}_q$ for computing the syndrome efficiently. Starting by sorting the vectors $\mathbf{a}_I$ by the first k entries, we then partition them accordingly; thus there are q^k partitions denoted $\mathcal{P}_j$, for $1 \leq j \leq q^k$. We can read the syndrome from its last $N-k$ entries directly if the vector $\mathbf{a}_I$ belongs to the partition with the first k entries all zero. Then we operate inductively. If we know one syndrome, we can compute another one in the same partition within $2(N-k)$ $\mathbb{Z}_q$ operations, or compute one in a different partition whose first k entries with distance 1 from that in the known partition within $3(N-k)$ $\mathbb{Z}_q$ operations. Suppose we have m_{dec} vectors to decode here (generally, the value m_{dec} is larger than q^k), then the complexity of this part is bounded by $(N-k)(2m_{dec}+q^k) < 3m_{dec}(N-k)$. Since the cost of adding error vectors for the codewords is $m_{dec}N$, we can give an upper bound for the decoding cost, which is roughly $4m_{dec}N$.

Concatenated Constructions. The drawback of the previous decoding strategy is that a large table is required to be stored with size exponential in $N-k$. On the other hand, there is an inherent memory constraint, i.e., $\mathcal{O}\left(q^b\right)$, when the size b is fixed, which dominates the complexity of the BKW-type algorithm.

We make use of a narrow sense concatenated code defined by direct summing several smaller linear codes to simplify the decoding procedure, when the decoding table is too large. This technique is not favored in coding theory since it diminishes the decoding capability, but it works well for our purpose.

5 Algorithm Description

We present a detailed description of the new algorithm in this section, containing five steps. This is illustrated in Algorithm 1 below.

5.1 Gaussian Elimination

The goal of this step is to transform the distribution of secret vector $\mathbf{s}$ to be that of the error (c.f. [4,23] for similar ideas). We refer to the full version for details on deriving the complexity of this step.

The complexity of this step is as follows,

$$C_0 = (m-n') \cdot (n+1) \cdot \lceil \frac{n'}{b-1} \rceil < m(n+1) \cdot \lceil \frac{n'}{b-1} \rceil, \tag{7}$$

where $n' = n - t_1 b$.

Algorithm 1. New LWE solving algorithm (main steps)

Input: Matrix $\mathbf{A}$ with n rows and m columns, received vector $\mathbf{z}$ of length m and algorithm parameters $t_1, t_2, b, n_{test}, l, d$

repeat
- Step 1: Gaussian elimination to change the distribution of the secret vector;
- Step 2: Use t_1 BKW steps to remove the bottom $t_1 b$ entries;
- Step 3: Use t_2 coded-BKW steps to remove the bottom n_{cod} entries;
- Step 4: **for** *every guess of the top* n_{top} *entries* **do**
  - Subspace Hypothesis Testing using an $[n_{test}, l]$ linear code and FFT;

until *acceptable hypothesis is found*

5.2 Standard BKW Reductions

The previously described coded-BKW in Sect. 4 introduces noise that grows with each iteration, so it makes sense to start with a number of pure BKW reductions. We start by performing t_1 standard BKW steps to balance the two noise parts, i.e., the noise increased by merging and the noise introduced by coding. This step zeros out the bottom $t_1 \cdot b$ bits. We now explain the details.

Given the output of the Gaussian elimination, i.e., $\hat{\mathbf{z}}$ and $\hat{\mathbf{A}} = (\mathbf{I}\mathbf{L}_0)$, we process only on the non-systematic part of $\hat{\mathbf{A}}$, denoted by $\mathbf{L}_0$. Similar as the other BKW procedures [7], in each step we sort the vector by the last b unprocessed entries and thus divide the total samples into at most $\frac{q^b-1}{2}$ classes. Then, we merge (adding or subtracting) those in the same class to zero the considered b entries, forming new samples as the input to the next BKW step, $\mathbf{L}_1, \mathbf{L}_2$, etc.

The output of this step is a matrix $(\mathbf{I}\mathbf{L}_t)$, where all $t_1 b$ last entries in each column of $\mathbf{L}_t$ are zero. Collecting the first $n - t_1 b$ rows of the matrix $\mathbf{L}_t$ and appending the identity matrix in front, we have a series of new LWE samples with dimension $n - t_1 b$. The complexity of this step is

$$C_1 = \sum_{i=1}^{t_1} (n + 1 - ib)(m - \frac{i(q^b - 1)}{2}). \tag{8}$$

5.3 Coded-BKW Reductions

We next continue to perform t_2 coded-BKW steps, in each of which an $[N_i, b]$ q-ary linear code is utilized. Here various rates are employed to equalize the error contribution per dimension. The code length N_i in the $(t_2 - i + 1)$th coded-BKW step is a function of a preset variance value σ^2_{set} which is determined by the error level introduced by the codes utilized in the last phase — subspace hypothesis testing. We know that in the final error expression there are $2^{t_2 - i + 1}$ error terms from the i-th coded BKW step. Thus, we have the following equation,

$$\sigma^2_{set} = \frac{2^i q^{2(1 - \frac{b}{N_i})}}{12}.$$

Thus, the code length N_i is chosen as,

$$N_i = \left\lfloor \frac{b}{1 - \frac{1}{2}\log_q(12 \cdot \frac{\sigma_{set}^2}{2^i})} \right\rfloor.$$

By n_{cod} we denote the total number of positions canceled by the coded-BKW steps, i.e., $n_{cod} = \sum_{i=1}^{t_2} N_i$. We denote the number of samples after the last coded-BKW step by M. Following Sect. 4.3, the decoding cost is upper bounded by

$$C_2' = \sum_{i=1}^{t_2} 4(M + \frac{i(q^b - 1)}{2})N_i,$$

where $(M + \frac{i(q^b-1)}{2})$ is the number of samples processed in the $(t_2 - i + 1)$-th step. Thus, the overall complexity of this step is

$$C_2 = C_2' + \sum_{i=1}^{t_2}(n_{top} + n_{test} + \sum_{j=1}^{i} N_j)(M + \frac{(i-1)(q^b - 1)}{2}). \tag{9}$$

5.4 Partial Guessing

The previous step outputs samples with smaller dimension but higher noise variance. In order to deal with the remaining unknowns in the secret $\hat{\mathbf{s}}$ vector, we use a combination of testing all values by guessing and performing a hypothesis test using an FFT.

In this step we perform a simple partial guessing technique, which balances the complexity of the previous steps and the later FFT testing phase. We exhaust the top n_{top} entries of $\hat{\mathbf{s}}$ with the absolute value less than d; thus there are $(2d+1)^{n_{top}}$ candidates. Thus, the complexity of this step is just that of updating the observed symbol, i.e.,

$$C_3 = Mn_{top}(2d + 1)^{n_{top}}. \tag{10}$$

The upcoming last step is performed for each such guess.

5.5 Subspace Hypothesis Testing

Here we generalize the subspace hypothesis testing technique first proposed in [22] to $\mathbb{Z}_q$ case, and then combine with Fast Fourier Transform to calculate the occurrences of different symbols in $\mathbb{Z}_q$ efficiently. This information would yield an optimal distinguisher with a small additional cost.

We use a polynomial in the quotient ring $\mathbb{Z}[X]/(X^q - 1)$ to record the occurrences. The modulus $(X^q - 1)$ is determined by the group property of $\mathbb{Z}_q$. We employ an $[n_{test}, l]$ systematic linear code, group the samples $(\hat{\mathbf{a}}_i', \hat{z}_i')$ from the previous steps in sets $L(\mathbf{c}_i)$ according to their nearest codewords and define the function $f_L^{\mathbf{c}_i}(X)$ as

$$f_L^{\mathbf{c}_i}(X) = \sum_{(\mathbf{a}_i', z_i') \in L(\mathbf{c}_i)} X^{\hat{z}_i' (\mathrm{mod}\ q)}.$$

Due to the systematic feature of the code utilized, we rewrite $f_L^{\mathbf{c}_i}(X)$ as a function of the information part $\mathbf{u}$ of the codeword $\mathbf{c}_i$, denoted by $h_{\mathbf{u}}(X) = f_L^{\mathbf{c}_i}(X)$, and later we exhaust all the q^l possible values of the vector $\mathbf{u}$. Define

$$H_{\mathbf{y}}(X) = \sum_{\mathbf{u}\in\mathbb{Z}_q^l} h_{\mathbf{u}}(X) \cdot X^{-\langle \mathbf{y},\mathbf{u}\rangle}.$$

Here we exhaust all candidates of $\mathbf{y} \in \mathbb{Z}_q^l$. Then, there exits a unique vector $\mathbf{y} \in \mathbb{Z}_q^l$, s.t., $\langle \mathbf{y}, \mathbf{u}\rangle = \langle \hat{\mathbf{s}}, \mathbf{c}_i\rangle$. For the right guess, the polynomial $H_{\mathbf{y}}(X)$ will record the occurrences of the error symbols which are discrete Gaussian distributed; otherwise, it should be uniformly distributed.

The calculation of the polynomial $H_{\mathbf{y}}(X)$ can be accelerated by Fast Fourier Transform. Let ω be a primitive q-th root of unity in the complex field $\mathbb{C}$. We can interpolate the polynomial $H_{\mathbf{y}}(X)$ if we know its q values at the q different points $(1, \omega, \omega^2, \ldots, \omega^{q-1})$ with complexity about $\mathcal{O}(q \log_2(q))$. Thus, the problem is transformed to a polynomial evaluation problem.

We first evaluate q^l polynomials $h_{\mathbf{u}}(X)$ on q different points $(1, \omega, \omega^2, \ldots, \omega^{q-1})$ with the complexity $\mathcal{O}\left(q^l \cdot q \log_2 q\right)$. Then with these values stored, we can evaluate the polynomial $H_{\mathbf{y}}(X)$ using q FFTs, each of which costs $\mathcal{O}\left(q^l \log_2(q^l)\right)$.

If the symbol occurrences are known, then we obtain the belief levels of all the candidates using a Neyman-Pearson test [15]. We choose the one with the highest rank and output it. This testing adds $\mathcal{O}\left(q^{l+1}\right)$ $\mathbb{Z}_q$-operations. Similar to that in the LPN case [22], recovering the remaining information can be done by iteratively employing this procedure to solve smaller LWE instances whose complexity is negligible compared to that of knowing the first part.

The employed $[n_{test}, l]$ linear code brings in an error with variance per dimension $\frac{q^{2(1-l/n_{test})}}{12}$. We denote it by σ_{set}^2, which is manipulated as a preset parameter in the previous coded-BKW phase to control the code sizes. As before, the decoding cost in this step is upper bounded by

$$C_4' = 4Mn_{test}.$$

We introduce a new notation $n_{tot} = n_{cod} + n_{test}$ to denote the total length of the subvectors affected by coding. The overall complexity of this step is given as

$$C_4 = C_4' + (2d+1)^{n_{top}}(C_{FFT} \cdot q^{l+1}(l+1)\log_2 q + q^{l+1}). \tag{11}$$

Here C_{FFT} is the constant before the complexity order of an FFT.

6 Analysis of the New Approach for BKW

We denote by $P(d)$ the probability that the absolute value of one guessed symbol $\hat{s}_i$ is smaller than d, where $\hat{s}_i \xleftarrow{\$} \mathcal{X}_\sigma$. Here we obtain a lower bound of $P(d)$ by ignoring the folding feature of the distribution as $P(d) > \mathrm{erf}(\frac{d}{\sqrt{2}\sigma})$, where erf is the error function $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$.

In the testing step, we preset a noise level $\gamma^2\sigma^2\sigma_{set}^2 n_{tot}$ to be the variance of the noise introduced by coding, and then compute the required number of samples to perform a successful distinguishing. The process may fail if the size of the information subvector to be tested, denoted $\hat{\mathbf{s}}_{test}$, is too large to distinguish. Thus we need a new notion, P_{test}, to denote the probability that the Euclidean length of $\hat{\mathbf{s}}_{test}$ is less than a preset value $\gamma\sqrt{n_{tot}}\sigma$. Using the following lemma from [28], which is a tail bound on discrete Gaussians, we can upper bound the failure probability by $(\gamma e^{\frac{1-\gamma^2}{2}})^{n_{tot}}$.

Lemma 1. *For any* $\gamma \geq 1$, $\Pr[||\mathbf{v}|| > \gamma\sigma\sqrt{n}; \mathbf{v} \xleftarrow{\$} D_{\mathbb{Z}^n,\sigma}] < (\gamma e^{\frac{(1-\gamma^2)}{2}})^n$.

Later we set the value γ to be 1.2. Then, the estimated success probability is larger than 97.5 % in most of the applications. We summarize our findings in the following theorem.

Theorem 1 (The Complexity of Algorithm 1). *Let* (n, q, σ) *be the parameters of the chosen* LWE *instance. Let* $d, t_1, t_2, b, l, n_{test}$ *be algorithm parameters. The number of* $\mathbb{Z}_q$ *operations required for a successful run of the new attack is*

$$C = \frac{C_0 + C_1 + C_2 + C_3 + C_4}{(P(d))^{n_{top}} \cdot P_{test}}, \tag{12}$$

with $C_0, \ldots, C_4$ *as in Eqs.* (7)–(11).

The required number of samples M *for testing is set to be*[3]

$$M = \frac{4\ln((2d+1)^{n_{top}} q^l)}{\Delta(\mathcal{X}_{\sigma_{final}} \| U)},$$

where U *is the uniform distribution in* $\mathbb{Z}_q$ *and* $\sigma_{final}^2 = 2^{t_1+t_2}\sigma^2 + \gamma^2\sigma^2\sigma_{set}^2 n_{tot}$. *Thus, the number of calls to the* LWE *oracle is*

$$m = \frac{(t_1 + t_2)(q^b - 1)}{2} + M.$$

Proof. The cost for one iteration is $C_0 + C_1 + C_2 + C_3 + C_4$, which should be divided by its expected success probability $(P(d))^{n_{top}} \cdot P_{test}$.

7 A Variant of Coded-BKW for Binary-LWE

We can derive an efficient algorithm for BINARY-LWE by modifying certain steps accordingly. First, the distribution of the information vector is already of small size in $\mathbb{Z}_q$; therefore we skip the Gaussian elimination step. In addition, since the prime q is a relatively large symbol, it is beneficial to replace the step of the FFT hypothesis testing by a simple step exhausting all the combinations of the top n_{top} entries, which are uniformly chosen from the binary set $\{0, 1\}^{n_{top}}$. The variant is similar to Algorithm 1, so we omit it here and refer the interested reader to the full version for details.

[3] The constant factor in the formula is chosen as 4. According to some estimates on linear and differential cryptanalysis (e.g., [6,33]), its failure probability is fairly low.

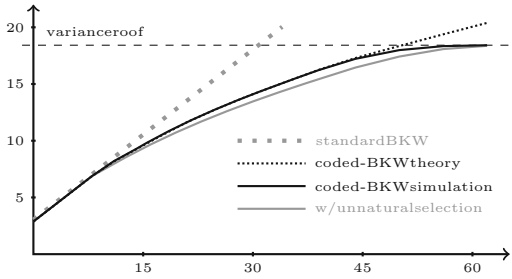

Fig. 1. Number of eliminated rows vs. $\log_2$ of error variance.Number of eliminated rows vs. $\log_2$ of error variance.

8 Simulation

We have performed simulations to support our theoretical results. A simulation with parameters $(q, \sigma, \#samples) = \left(2053, 2.70, 2^{25}\right)$ is shown in Fig. 1, plotting the number of eliminated rows vs. $\log_2$ of the variance of the samples errors. Four standard 2-row BKW steps were used initially, followed by three iterations each of [3,2]-, [4,2]-, [5,2]- and [6,2]-coding steps. The dashed horizontal line shows the variance of the corresponding uniform distribution (variance roof) of the errors, setting an upper bound for variance in simulations. The four curves show the performances of 2-step BKW (theoretical), theoretical coded-BKW (according to Sect. 6), coded-BKW simulation, and coded-BKW simulation when employing the unnatural selection heuristic (see [3]).

It is clear that coded-BKW significantly outperforms plain BKW. Furthermore, it can be seen that the developed theoretical estimations for coded-BKW very closely match actual simulation performance.

Last but not least, the unnatural selection heuristic can be employed by producing more samples, but retaining only the ones with the smallest coding errors. Instead of producing 2^{25} samples, 2^{27} were produced at each step. There is a clear gain in variance performance, and that gain is even larger when the sample factor is increased. These results will be detailed in the full version.

9 Summary of Results

We now present numerical results, as shown in Tables 1 and 2, using the new algorithms to solve the LWE and BINARY-LWE problems for various parameter settings, including instances from Regev's cryptosystem [31] or from Lindner and Peikert's paper [25]. As in [17], we consider operations over $\mathbb{C}$ to have the same complexity as the operation in $\mathbb{Z}_q$, and set C_{FFT} to be 1, which is the best we can obtain for an FFT. We also set $\gamma = 1.2$ and $d = 3\sigma$.

As in [1], we apply the new method to the instances proposed in a somewhat homomorphic encryption scheme [2], which can be considered as LWE instances using linearization. Our method yields substantial improvements in all cases

and especially solves an instance with the number of variables in the linearized system $n = 153$ (targeting 128-bit security [1]), in about 2^{119} bit operations, thereby breaking the claimed security level.

We present here additional information about the comparisons in Tables 1 and 2. Firstly, only the new algorithms and the algorithm proposed in [17] are key-recovery attacks; all the others belong to the class of distinguishing attacks. Secondly, the counterpart proposed by Albrecht et al. [3] is the version without unnatural selection, since we can also improve our algorithm by this heuristic. Thus, we accelerate the BKW-type BINARY-LWE solver by a factor of almost 2^{20}, for the toy instance $n = 128$ in Regev's parameter setting. Last, we adopt the estimating model in [1,3] using data from the implementations in [11,25,26] to evaluate the performance of the lattice reduction distinguisher, when LWE is reduced to SIS. We refer the interested readers to these two papers for details.

When reducing LWE to BDD, also named "Decode" in [25], Lindner and Peikert reported the running time of this attack on two LWE instances. Albrecht et al. [1] multiplied the time by the clock speed of the CPU used, compared with their BKW variant, and finally reached the conclusion that the BDD approach would yield substantially lower complexity. Specifically, their estimation about this "Decode" approach on one (with parameter $(136, 2003, 5.19)$) of the two instances is about $2^{91.4}$ $\mathbb{Z}_q$ operations. We obtain a much better time complexity of about $2^{80.6}$ operations over $\mathbb{Z}_q$, when applying Algorithm 1 to this instance.

As RING-LWE is a sub-problem of LWE, the new algorithm can be employed to attack some recent RING-LWE-based cryptosystems [16,21,32]. We solve the underlying RING-LWE $(256, 7681, 4.51)$ and RING-LWE $(512, 12289, 4.86)$ instantiations in 2^{123} and 2^{225} bit-operations, respectively, thereby breaking the claimed 128-bit and 256-bit security levels.

10 Conclusion

We have proposed a new algorithm to solve the LWE problem by modifying the steps of the BKW algorithm using lattice codes. Our algorithm outperforms the previous BKW variants for all instantiations we considered and also all the lattice reduction approaches from some size of instances and onwards. To the best of our knowledge, it is the best LWE solver when the dimension n is large enough and it seems to cover the choices of today's and future security levels. Another application is that it outperforms all the other approaches drastically on the BINARY-LWE problem.

References

1. Albrecht, M.R., Cid, C., Faugère, J.C., Fitzpatrick, R., Perret, L.: On the Complexity of the BKW Algorithm on LWE. Desig. Codes Crypt. **74**, 1–30 (2013)
2. Albrecht, M.R., Farshim, P., Faugère, J.-C., Perret, L.: Polly cracker, revisited. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 179–196. Springer, Heidelberg (2011)

3. Albrecht, M.R., Faugère, J.-C., Fitzpatrick, R., Perret, L.: Lazy modulus switching for the BKW algorithm on LWE. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 429–445. Springer, Heidelberg (2014)
4. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)
5. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 403–415. Springer, Heidelberg (2011)
6. Baignères, T., Junod, P., Vaudenay, S.: How far can we go beyond linear cryptanalysis? In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 432–450. Springer, Heidelberg (2004)
7. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM **50**(4), 506–519 (2003)
8. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012)
9. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing, STOC 2013, pp. 575–584. ACM (2013)
10. 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)
11. Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
12. Cohen, G., Honkala, I., Litsyn, S., Lobstein, A.: Covering Codes, vol. 54. Elsevier, Amsterdam (1997)
13. Conway, J., Sloane, N.: Voronoi regions of lattices, second moments of polytopes, and quantization. IEEE Trans. Inf. Theory **28**(2), 211–226 (1982)
14. Conway, J.H., Sloane, N.J.A., Bannai, E., Leech, J., Norton, S., Odlyzko, A., Parker, R., Queen, L., Venkov, B.: Sphere Packings, Lattices and Groups, vol. 3. Springer, New York (1993)
15. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley, New York (2012)
16. De Clercq, R., Roy, S.S., Vercauteren, F., Verbauwhede, I.: Efficient software implementation of Ring-LWE Encryption. In: Design, Automation and Test in Europe (DATE 2015) (2015)
17. Duc, A., Tramèr, F., Vaudenay, S.: Better algorithms for LWE and LWR. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 173–202. Springer, Heidelberg (2015)
18. Erez, U., Litsyn, S., Zamir, R.: Lattices which are good for (almost) everything. IEEE Trans. Inf. Theory **51**(10), 3401–3416 (2005)
19. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009)
20. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013)

21. Göttert, N., Feller, T., Schneider, M., Buchmann, J., Huss, S.: On the design of hardware building blocks for modern lattice-based encryption schemes. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 512–529. Springer, Heidelberg (2012)
22. Guo, Q., Johansson, T., Löndahl, C.: Solving LPN using covering codes. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 1–20. Springer, Heidelberg (2014)
23. Kirchner, P.: Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377 (2011)
24. Levieil, É., Fouque, P.-A.: An improved LPN algorithm. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 348–359. Springer, Heidelberg (2006)
25. Lindner, R., Peikert, C.: Better key sizes (and Attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
26. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: an update. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 293–309. Springer, Heidelberg (2013)
27. Loeliger, H.A.: Averaging bounds for lattices and linear codes. IEEE Trans. Inf. Theory **43**(6), 1767–1773 (1997)
28. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012)
29. 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)
30. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, pp. 333–342. ACM (2009)
31. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 34:1–34:40 (2009)
32. Roy, S.S., Vercauteren, F., Mentens, N., Chen, D.D., Verbauwhede, I.: Compact Ring-LWE cryptoprocessor. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 371–391. Springer, Heidelberg (2014)
33. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. J. Crypt. **21**(1), 131–147 (2008)
34. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 288. Springer, Heidelberg (2002)
35. Zamir, R., Feder, M.: On lattice quantization noise. IEEE Trans. Inf. Theory **42**(4), 1152–1159 (1996)

http://www.springer.com/978-3-662-47988-9

Advances in Cryptology -- CRYPTO 2015
35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I
Gennaro, R.; Robshaw, M. (Eds.)
2015, XVIII, 787 p. 108 illus., Softcover
ISBN: 978-3-662-47988-9