REVIEW

Open Access

Security estimation of LWE via BKW algorithms



Yu Wei^{1,2}, Lei Bi^{1,2*}, Xianhui Lu^{1,2} and Kunpeng Wang^{1,2}

Abstract

The Learning With Errors (LWE) problem is widely used in lattice-based cryptography, which is the most promising post-quantum cryptography direction. There are a variety of LWE-solving methods, which can be classified into four groups: lattice methods, algebraic methods, combinatorial methods, and exhaustive searching. The Blum–Kalai–Wasserman (BKW) algorithm is an important variety of combinatorial algorithms, which was first presented for solving the Learning Parity With Noise (LPN) problem and then extended to solve LWE. In this paper, we give an overview of BKW algorithms for solving LWE. We introduce the framework and key techniques of BKW algorithms and make comparisons between different BKW algorithms and also with lattice methods by estimating concrete security of specific LWE instances. We also briefly discuss the current problems and potential future directions of BKW algorithms.

Keywords Post-quantum cryptography, Learning with errors problem, Lattice-based cryptography, Security estimation

Introduction

The learning with errors (LWE) problem, introduced by Regev (2005), has received widespread attention over the last decade. It is one of the most important problems in lattice-based cryptography. The schemes based on LWE and its variants have developed rapidly and are regarded as one of the most promising routes for the standardization of post-quantum cryptography. Due to its efficiency, versatility, and theoretical reduction to standard lattice problems, the LWE problem has various applications. For instance, significant research progress has been made in Attribute-based Encryption (ABE) (Boyen 2013; Brakerski and Vaikuntanathan 2016), Fully Homomorphic Encryption (FHE) (Gentry 2009; Brakerski and Vaikuntanathan 2014), Function Encryption (FE) (Agrawal et al. 2015; Goldwasser et al. 2014), key exchange protocols

Lei Bi

bilei121@outlook.com

(Ding et al. 2012; Alkim et al. 2015), and digital signatures (Abdalla et al. 2012; Güneysu et al. 2012) based on the LWE problem.

Definition 1 (Regev 2005) Let *n*, *q* be positive integers, and χ be an error distribution over \mathbb{Z} . **s** is a randomly uniform secret vector in \mathbb{Z}_q^n . Choose $\mathbf{a} \in \mathbb{Z}_q^n$ randomly and $e \in \mathbb{Z}$ according to χ , return $(\mathbf{a}, z) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e \mod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ as the samples from distribution $L_{\mathbf{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The Search-LWE aims to identify **s** given some samples. The Decision-LWE aims to distinguish whether the samples are from $L_{\mathbf{s},\chi}$ or a uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The security of LWE is a prominent area of publickey cryptographic research. A cryptanalytic treatment of LWE includes a concrete security estimation and an extrapolation from asymptotic complexity to cryptographic security level. The concrete security estimation is to calculate the number of operations of certain algorithms for solving LWE (Albrecht et al. 2015a). While the asymptotic analysis describes the performance of the



© The Author(s) 2023. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

^{*}Correspondence:

 ¹ SKLOIS, Institute of Information Engineering, CAS, Beijing, China
 ² School of Cyber Security, University of Chinese Academy of Sciences,

Beijing, China

algorithms by 2^{cn} , where *n* is the dimension of LWE (Herold et al. 2018).

Algorithms for solving LWE

There are several LWE-solving algorithms in the literature, which can be divided into four groups: methods based on solving lattice problems (Chen and Nguyen 2011; Schnorr and Euchner 1994; Lindner and Peikert 2011; Liu and Nguyen 2013; Albrecht and Fitzpatrick 2013; Micciancio and Regev 2009), algebraic methods (Arora and Ge 2011), combinatorial methods (Blum et al. 2003; Dong et al. 2021), and exhaustive searching (Bi et al. 2019).

Due to the strong connection with lattice problems, the most common method is to reduce LWE to a lattice problem and then settle it with lattice reduction (Chen and Nguyen 2011; Schnorr and Euchner 1994). For example, Search-LWE can be directly reduced to the Bounded Distance Decoding (BDD) problem (Lindner and Peikert 2011; Liu and Nguyen 2013). The BDD problem can also be reduced to the unique Shortest Vector Problem (uSVP) (Albrecht and Fitzpatrick 2013). Alternatively, the LWE problem can be rewritten as a Short Integer Solution (SIS) problem, which aims to find a short vector in a dual lattice (Micciancio and Regev 2009). Although this sort of approach does not require exponential samples, there seems to be some ambiguity regarding the complexity of higher dimensions.

The algebraic approach was proposed by Arora and Ge (2011), which transforms solving LWE into solving non-linear equations. This approach can solve the LWE problem in a sub-exponential time when the Gaussian distribution is narrow sufficiently, i.e., $\sigma < \sqrt{n}$ where σ is the standard deviation of the Gaussian distribution. Otherwise, it takes fully exponential time. However, in practice, this approach is much more expensive than other approaches for the parameters commonly considered in cryptographic applications (Albrecht and Faugre 2012).

Combinatorial algorithms for solving LWE generally use the well-known BKW (Blum et al. 2003) algorithm, which is the focus of this article, as a foundation. The Meet-In-The-Middle (MITM) attack also belongs to combinatorial algorithms (Dong et al. 2021). These combinatorial approaches have the benefit of being standardized in their complexity analysis, allowing us to obtain explicit complexity values for different instantiations of LWE. The disadvantage of these approaches is that their memory requirements are often on the same scale as their time complexity.

BKW algorithms

The BKW algorithm is similar to Wagner's (2002) generalized birthday approach and was originally proposed to solve the LPN problem. Later, a lot of advancements for BKW solving LPN appeared (E Levieil 2006; Kirchner 2011; Guo et al. 2014; Zhang et al. 2016; Bogos and Vaudenay 2016; Bogos et al. 2015). Subsequently, together with these fresh techniques, BKW algorithms are extended to solving LWE.

When solving LWE, the most trivial strategy is to exhaust all vectors \mathbf{a} and find the samples where all positions except one are zero. Thus, it is possible to get the corresponding position component of the secret vector \mathbf{s} . However, the time and sample complexities of successfully solving LWE with this approach are both super-exponential.

Based on the above idea, the first BKW algorithm (we refer to it as Plain BKW algorithm in this paper) using the sort-and-match technique for solving LWE was proposed by Albrecht et al. (2015b). The Plain BKW algorithm has three stages: sample reduction, hypothesis testing, and back substitution.

Sample reduction partitions all vectors \mathbf{a} into 'blocks' which are then sorted, after which new samples are created by finding collisions in the 'blocks' and then reducing these positions to zero by matching (i.e., adding or subtracting). The new vectors \mathbf{a} obtained in this way are all zero except for one or two positions, at the cost of increased noise.

Hypothesis testing aims to distinguish the correct guess of the secret sub-vector from incorrect ones. If the guessed value is correct, the distribution of the observed noise elements will follow a Gaussian distribution. Otherwise, these values will be uniformly random.

Back substitution allows the operation to be repeated on a smaller LWE instance after we have obtained some information about the secret vector.

The BKW algorithm for solving LWE has undergone numerous significant developments in sample reduction and hypothesis testing, which are at the heart of BKW algorithms, over the last ten years. We will introduce them as follows, and an overview is shown in Fig. 1.

Developments of sample reduction

There are currently four classes of BKW algorithms optimizing the sample reduction stage: LMS-BKW (Albrecht et al. 2014), Coded-BKW (Guo et al. 2015), Sieve-Coded-BKW (Guo et al. 2017, 2019; Mårtensson 2019) and BKW-FWHT-SR (Budroni et al. 2021), which use different techniques.

The LMS-BKW algorithm, introduced by Albrecht et al. (2014), uses the lazy modulus switching technique. It chooses a modular p < q and searches for collisions by only taking into account the top $\log_2 p$ bits of each component in the 'blocks'. Thus, it does not reduce the collisions to zero like Plain BKW but to a smaller



Fig. 1 The progress of BKW algorithms for solving LWE

value, which will accumulate as the number of iterations increases. This algorithm can remove more components in each step than Plain BKW, but the overall noise distribution of the final samples is not uniform.

The Coded-BKW algorithm, proposed by Guo et al. (2015), uses linear lattice codes to map the 'blocks' of **a** vectors into the nearest codeword in a lattice code. If any pairs of the 'blocks' in **a** map to the identical codeword, these two vectors are merged to produce a new sample with this block reduced.

Same as LMS-BKW, each iteration of Coded-BKW does not completely reduce the block to zero, which introduces some noise. With proper parameter selection, the coding noise can be maintained low enough that it has no discernible effect on the final noise. By gradually increasing the step size, the issue of imbalanced final noise of LMS-BKW can be fixed.

The Sieve-Coded-BKW algorithm (Guo et al. 2017) is a combination of Coded-BKW with lattice sieving. By using sieving, this algorithm ensures that the noise from reduced components does not increase dramatically, which addresses the issue of the increasing coding noise in Coded-BKW. Each iteration is divided into two steps. The first step is to find vector pairs that can be matched to reduce the size of some positions. To ensure that the size of the reduced vector positions is approximately the same as the already reduced vector positions, the second step uses sieving to cover all components from all preceding steps. Afterwards, the γ -Sieve-Coded-BKW algorithm (Guo et al. 2019) takes the same value of reduction factor γ in each iteration, where the reduction factor is not limited to 1 like Sieve-Coded-BKW but belongs to $(0, \sqrt{2})$. After that, an improved γ_i -Sieve-Coded-BKW algorithm (Mårtensson 2019) takes different γ_i in each iteration, which outperforms the former two Coded-BKW with sieving in asymptotic complexity.

The BKW-FWHT-SR algorithm, proposed by Budroni et al. (2021), uses a modified reduction step of lazy modulus switching, which was called Smooth-LMS. This new approach is to partially reduce one additional component after reducing a given number of components using easy LMS. The sample reduction stage balances the complexity between various reduction steps. As a result, it can reduce more components overall with the same time and memory complexity in each iteration. The complexity of this algorithm outperforms all other previous methods.

Developments of hypothesis testing

After sample reduction, we guess the secret vector partially and distinguish the correct one from others by hypothesis testing. Initially, the hypothesis testing stage used log-likelihood estimation. The tool used to improve this stage of the BKW algorithm is mainly the Fast Fourier Transform (FFT) technique. The FFT-BKW algorithm proposed by Duc et al. (2015) is the first BKW algorithm to use the FFT-distinguisher. In contrast to Plain BKW (Albrecht et al. 2015b), FFT-BKW eliminates integrals that are challenging to analyze in the final complexity.

Later, the Coded-BKW algorithm (Guo et al. 2015) proposed the subspace hypothesis testing in \mathbb{Z}_q , which is extended from the \mathbb{Z}_2 case. Then this method combined it with FFT to efficiently record the occurrences of the error symbols in \mathbb{Z}_q . If we guess correctly, these error terms follow a Gaussian distribution; otherwise, they are uniformly random.

Afterwards, the pruned FFT algorithm (Guo et al. 2021) and the BKW-FWHT-SR algorithm (Budroni et al. 2021) used pruned FFT distinguisher and the Fast

Walsh-Hadamard Transform (FWHT) distinguisher to optimize the hypothesis testing separately. The pruned FFT distinguisher works by limiting the number of hypotheses, and the FWHT allows for a more accurate distinction with a larger noise level.

Contribution

We provide an overview of BKW algorithms as follows:

- We review the improvements of the sample reduction stage and the hypothesis testing stage, including some technologies such as lazy modulus switching, linear lattice codes, sieving, FFT, and so on.
- (2) We estimated the concrete security of specific LWE instances using various BKW algorithms and lattice-based algorithms and present comparisons of them.
- (3) We discuss the current problems and potential future directions of BKW algorithms.

Organization

The remaining part of this paper is arranged as follows. "Preliminaries" section states some necessary background. The framework of Plain BKW is discussed in "Plain BKW" section. We demonstrate the improvements of the sample reduction stage and the hypothesis testing stage in "Improvements of sample reduction in BKW algorithms" section and "Improvements of hypothesis testing in BKW algorithms" section, respectively. In "Comparisons" section we make a comparison of BKW algorithms by presenting the features and concrete security estimation results on LWE instances by using them. Finally, we conclude this paper in "Conclusion" section.

Preliminaries

Notations

In Euclidean space \mathbb{R}^n with *n* dimensions, the L_2 -norm of a vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is defined as the square root of the sum of the squares of its components: $||\mathbf{x}|| = \sqrt{x_1^2 + \dots + x_n^2}$. Denoted by $||\mathbf{x} - \mathbf{y}||$ the Euclidean distance between two vectors \mathbf{x} and \mathbf{y} in \mathbb{R}^n . Given a vector \mathbf{x} and start counting at zero, $\mathbf{x}_{(a,b)}$ represents the vector $(\mathbf{x}_{(a)}, \dots, \mathbf{x}_{(b-1)})$. $[\cdot] : \mathbb{R} \to \mathbb{Z}$ denotes the rounding function that rounds to the closest integer. The set of integers in $\left[-\frac{q-1}{2}, \frac{q-1}{2}\right]$ represents elements in \mathbb{Z}_q .

Discrete Gaussian distribution

Denote the discrete Gaussian distribution over \mathbb{Z} with mean 0 and variance σ^2 as $D_{\mathbb{Z},\sigma}$. The $\chi_{\sigma,q}$ distribution over \mathbb{Z}_q (often be written as χ_{σ}) with variance σ^2 is obtained by folding $D_{\mathbb{Z},\sigma} \mod q$, namely adding up the

probability mass function value for each residue class mod *q* across all integers. Let the noise level be represented by α , where $\alpha = \sigma/q$.

Even while the discrete Gaussian distribution often does not precisely take the typical features of the continuous one, we will still be able to use the continuous features since they will be close enough. If two independent distributions *X* is taken from χ_{σ_1} and *Y* is taken from χ_{σ_2} , then their sum X + Y is taken from $\chi_{\sqrt{\sigma_1^2 + \sigma_2^2}}$.

The LWE problem reformulated

The LWE problem can be reformulated as a decoding problem. Here are *m* samples

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

selected from $L_{\mathbf{s},\chi}$, where $\mathbf{a}_i \in \mathbb{Z}_q^n$, $z_i \in \mathbb{Z}_q$. Write $\mathbf{y} = (y_1, y_2, \dots, y_m) = \mathbf{s}\mathbf{A}$ and $\mathbf{z} = (z_1, z_2, \dots, z_m)$. Therefore, $\mathbf{A} = [\mathbf{a}_1^T \mathbf{a}_2^T \cdots \mathbf{a}_m^T]$ and $\mathbf{z} = \mathbf{s}\mathbf{A} + \mathbf{e}$, where $z_i = y_i + e_i = \langle \mathbf{s}, \mathbf{a}_i \rangle + e_i$ and $e_i \leftarrow \chi_\sigma$ is error. The matrix \mathbf{A} is responsible for generating a linear code in the field of \mathbb{Z}_q and \mathbf{z} represents the received message. The task of discovering \mathbf{s} is to find the codeword $\mathbf{y} = \mathbf{s}\mathbf{A}$, where the distance between \mathbf{z} and \mathbf{y} is smallest.

The transformation of secret distribution

If the secret vector **s** is uniformly random, a transformation (Kirchner 2011; Applebaum et al. 2009) can be used to guarantee that **s** follows the noise distribution χ_{σ} .

Through Gaussian elimination, we first transform $\mathbf{A}_{n \times m}$ into systematic form. Suppose that the first *n* columns of \mathbf{A} are linearly independent and denoted by the matrix \mathbf{A}_0 . Write $\mathbf{D} = \mathbf{A}_0^{-1}$ and $\hat{\mathbf{s}} = \mathbf{s}\mathbf{D}^{-1} - (z_1, z_2 \cdots, z_n)$. Thus, we can get a similar problem that $\hat{\mathbf{A}} = (\mathbf{I}, \hat{\mathbf{a}}_{n+1}^T, \hat{\mathbf{a}}_{n+2}^T, \dots, \hat{\mathbf{a}}_m^T)$, where $\hat{\mathbf{A}} = \mathbf{D}\mathbf{A}$. And then calculate

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

By this transformation, each component in **s** is distributed according to χ_{σ} , which makes sense to some famous reduction algorithms for solving LWE.

The sieving algorithm

In 2001, Ajtai–Kumar–Sivakumar proposed an algorithm for the Shortest Vector Problem (SVP), which was called sieving. Assume there is a large list \mathcal{L} with many short lattice vectors. The main goal of sieving is to efficiently find a vector $\mathbf{x} \in \mathcal{L}$ that is the closest to a vector $\mathbf{y} \in \mathcal{L}$. By addition or subtraction, we can get many pairs of vectors $\mathbf{x}, \mathbf{y} \in \mathcal{L}$ that satisfy $||\mathbf{x} \pm \mathbf{y}|| \le \max\{||\mathbf{x}||, ||\mathbf{y}||\}$. After repeating this reduction for polynomial times, we can get the shortest vector with a high probability. The most asymptotically efficient solution is Locality Sensitive Filtering (LSF) (Becker et al. 2016) which was used in conjunction with the BKW algorithms.

Plain BKW

The Plain BKW (Albrecht et al. 2015b) is the first BKW algorithm for solving LWE. It serves as the framework and foundation for subsequent BKW algorithms. Now we provide a detailed explanation of this algorithm, specifically focusing on the sample reduction and hypothesis testing stage, which are the main targets of subsequent optimization.

Intuition

When solving the LWE problem, we need to get some information about the secret vector **s**. A trivial method is to find sample vectors **a** where all positions except one are zero. For example, if we get some sample vectors like

$$\mathbf{a}_1 = (* \ 0 \ 0 \cdots 0 \ 0), \\ \mathbf{a}_2 = (0 \ * \ 0 \cdots 0 \ 0), \\ \cdots \\ \mathbf{a}_k = (0 \ 0 \cdots 0 \ 0 \ *),$$

we can recover some positions of secret **s** by solving the corresponding LWE equations.

However, this idea needs a large number of samples. The time and sample complexities of successfully finding the secret vector with it are both super-exponential.

Sample reduction

Based on the above idea, the BKW algorithm uses the sort-and-match technique to produce new samples during the sample reduction stage. Given *m* samples $(\mathbf{a}_i, z_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, we sort them into different groups based on the *b* positions of each vector **a**. If the *b* positions are eliminated when vectors **a** are matched (added/subtracted), such pairs of samples will be in the same group. For instance, samples $(\pm[\mathbf{a}_{10}, \mathbf{a}_{11}], z_1)$ and $(\pm[\mathbf{a}_{20}, \mathbf{a}_{21}], z_2)$ are in the same category if they can be added/subtracted to obtain

$$(\underbrace{[0\cdots 0}_{b}*\cdots*], z_1\pm z_2).$$

The resulting error is $e_{1,2} = e_1 + e_2$, which follows the distribution $\chi_{\sqrt{2}\sigma}$. After repeating the reduction process *t* times, we will get a new LWE instance whose dimension is $n - t \cdot b$, and the error follows the distribution $\chi_{\sqrt{2}t_{\sigma}}$.

The Plain BKW algorithm defines positive integers $b \le n$, $a := \lceil n/b \rceil$. We denote by $B_{\mathbf{s},\chi,l}$ the oracles that output samples whose first $b \cdot l$ positions of **a** are zero.

- For l = 0, $B_{\mathbf{s},\chi,0}$ is equivalent to $L_{\mathbf{s},\chi}$.
- For $1 \le l \le a$, $B_{\mathbf{s},\chi,l}$ is created from $B_{\mathbf{s},\chi,l-1}$. By constantly checking $B_{\mathbf{s},\chi,l-1}$, one can get at most $(q^b 1)/2$ samples (\mathbf{a}, z) with different non-zero values for *b* components of **a** and put them in the table T^l . Choose a new sample (\mathbf{a}', z') from $B_{\mathbf{s},\chi,l-1}$. As long as the absolute values of *b* components of \mathbf{a}' are matched by the *b* components of vector **a** in T^l , compute $(\mathbf{a}' \pm \mathbf{a}, z' \pm z)$ as a new sample from $B_{\mathbf{s},\chi,l}$. If the *b* components of \mathbf{a}' are already zero, compute (\mathbf{a}', z') as a new sample from $B_{\mathbf{s},\chi,l}$.

The process of recursively generating oracles $B_{\mathbf{s},\chi,l}$ is the process of sample reduction in Plain BKW.

Hypothesis testing

Assume that after *t* reduction processes, the vectors **a** are all zero except for the *d* positions, $0 \le d \le b$. Then we have samples

$$z_i = \sum_{j=1}^d a_{i_j} \cdot s_j + e_i \Leftrightarrow e_i = z_i - \sum_{j=1}^d a_{i_j} \cdot s_j,$$

where e_i follow the Gaussian distribution $\chi_{\sqrt{2^t}\sigma}$.

We then perform hypothesis testing on the unknown sub-vector $\mathbf{s}' = (s_1, s_2, \ldots, s_d)$ of the secret to distinguish the correct one among the q^d candidates. The error follows $\chi_{\sqrt{2^t}\sigma}$ if we guess \mathbf{s}' correctly, otherwise it is uniform.

The Plain BKW algorithm uses the log-likelihood ratio, which is the most effective test for determining whether samples belong to one of two given distributions according to the Neyman–Pearson Lemma (Neyman and Pearson 1933). The detailed formula is given in "Optimal distinguisher" section.

Back substitution

Note that the noiseless equations in linear systems are solved by converting them into triangles, locating an optimal solution, and then expanding this solution via back substitution. Using this terminology from linear algebra, the back substitution of BKW means the entire process can be carried out on a smaller LWE problem.

We employ back substitution when given the candidate answer to \mathbf{s}' that is very likely to be accurate. Following back substitution, the BKW algorithm can be restarted at the first stage. We then need *m* new samples that are reduced utilizing the modified tables and conduct hypothesis testing on them in order to find the next *d* positions of \mathbf{s} .

The complexity of plain BKW

Lemma 1 (Search-LWE, Albrecht et al. 2015b) Given m LWE samples $(\mathbf{a}_i, z_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, let $0 < b \leq n$, $d \leq b$, $a = \lceil n/b \rceil$, q is a prime. Then, the computational cost of Plain BKW for recovering **s** with success probability $0 < \varepsilon < 1$ is $\left((n+1) \cdot \frac{(a-1)a}{2} - \frac{(a-1)ab}{4} - \frac{b}{6}(1)\right)$ $\frac{1}{2}(a-1) + \frac{3}{2}(a-1)^2 + (a-1)^3\right) \cdot \left(\frac{q^b-1}{2}\right)$ operations in \mathbb{Z}_q to create the elimination tables,

$$\frac{\left|\frac{n}{d}\right|+1}{2} \cdot \frac{q^d}{q^d-1} \cdot m \cdot \left((n+2) \cdot \frac{a}{2}\right)$$

operations in \mathbb{Z}_q to create samples for hypothesis testing. For the hypothesis testing stage

$$(m \cdot q^d) \cdot \left\lceil \frac{n}{d} \right\rceil$$

operations in \mathbb{Z}_q are needed and

$$a \cdot d \cdot \lceil \frac{q^b}{2} \rceil \cdot \left(\lceil \frac{n}{d} \rceil + 1 \right)$$

operations in \mathbb{Z}_q for back substitution. Furthermore,

$$\lceil \frac{q^b}{2} \rceil \cdot a + m \cdot \lceil \frac{n}{d} \rceil \cdot \frac{q^d}{q^d - 1}$$

calls to $L_{\mathbf{s},\chi}$ and storage for

$$a \cdot \frac{q^b}{2} \cdot \left(1 + n - \frac{a-1}{2} \cdot b\right)$$

elements in \mathbb{Z}_q are needed.

Improvements of sample reduction in BKW algorithms

The core idea of sample reduction is to sort and match samples to eliminate some components of vectors **a** to zero. In order to optimize this stage, several new types of elimination methods have emerged. In this section, we will show the enhancements of the sample reduction stage in different BKW algorithms, including the LMS-BKW algorithm, the Coded-BKW algorithm, three types of Sieve-Coded-BKW algorithm, and the BKW-FWHT-SR algorithm. Finally, we will give a high-level comparison of them.

The LMS-BKW algorithm

Since the complexity of the BKW algorithm is essentially dependent on q^b and b doesn't depend on q, applying modulus switching may be intuitively expected to reduce

the complexity. Based on this idea, the LMS-BKW algorithm (Albrecht et al. 2014) can be viewed as a hybrid of Plain BKW and lazy modulus switching, which means changing to a lower precision when necessary rather than applying modulus switching in a 'one shot'.

The sample reduction stage of LMS-BKW only searches for collisions within the *b* components of each vector **a** by only taking into account the most significant $\log_2 p$ bits of \mathbb{Z}_q , where p < q is a positive integer. If such a collision is found, combine the colliding samples to eliminate the most significant $\log_2 p$ bits of the *b* components. This stage can also be seen as the process of recursively constructing several oracles $B_{\mathbf{s},\chi}(b, l, p)$ like the Plain BKW in Albrecht et al. (2015b), where $b \leq n$, $a := \lceil n/b \rceil$, $0 \leq l \leq a$.

- For l = 0, oracle $B_{\mathbf{s},\chi}(b, 0, p)$ is equivalent to $L_{\mathbf{s},\chi}$.
- For $1 \leq l \leq a$, $B_{\mathbf{s},\chi}(b,l,p)$ is created from $B_{\mathbf{s},\chi}(b,l-1,p)$. Constantly checking $B_{\mathbf{s},\chi}(b,l-1,p)$ can get at most $(q^b-1)/2$ samples (\mathbf{a},z) with different non-zero vectors $\lfloor p/q \cdot \mathbf{a}_{(b\cdot l-b,b\cdot l)} \rfloor$ and then put them in the table T^l . Choose a new sample (\mathbf{a}',z') from $B_{\mathbf{s},\chi}(b,l-1,p)$, as long as $\lfloor p/q \cdot \mathbf{a}'_{(b\cdot l-b,b\cdot l)} \rfloor$ (resp. their negation) matches $\lfloor p/q \cdot \mathbf{a}_{(b\cdot l-b,b\cdot l)} \rfloor$ in T^l , compute $(\mathbf{a}' \pm \mathbf{a}, z' \pm z)$ as a new sample from $B_{\mathbf{s},\chi}(b,l,p)$. If $\lfloor p/q \cdot \mathbf{a}'_{(b\cdot l-b,b\cdot l)} \rfloor$ is already zero, compute (\mathbf{a}',z') as a new sample from $B_{\mathbf{s},\chi}(b,l,p)$.

After repeating the iteration for *a* times, LMS-BKW has produced new samples in the format of $(\tilde{\mathbf{a}}, \tilde{z} = \langle \tilde{\mathbf{a}}, \mathbf{s} \rangle + \tilde{e})$, where $\tilde{\mathbf{a}}$ is short enough, \tilde{e} follows the Gaussian distribution $\chi_{\sqrt{2^a}\sigma}$. Assume that $|\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle| \approx \sqrt{2^a}\sigma$ to balance the increased initial noise \tilde{e} and the contribution of $|\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle|$ which is called the rounding noise.

To further reduce the size of the whole noise, a preprocessing step called unnatural selection was introduced. When finding collisions in positions of vector **a** with index $l \cdot b \leq j \leq l \cdot b + b$, select the samples with the smallest values in positions with index $j < b \cdot l$. As a result, the whole noise size of new samples can be further reduced.

The overall complexity of LMS-BKW and the behaviors of solving LWE via Plain BKW, LMS-BKW, BKZ with modulus switching, and MITM strategy can be found in Albrecht et al. (2014). Under their parameter settings, the LMS-BKW algorithm yields the best results when compared to other alternatives. However, LMS-BKW is based on many unproven assumptions, although they seem sound. Thus, it is appropriate to view their estimations as heuristics. Therefore, confirming these hypotheses is a viable direction for future research.

The Coded-BKW algorithm

The Coded-BKW algorithm (Guo et al. 2015) employs linear lattice codes to cancel more positions in each reduction step, a task that LMS-BKW fails to accomplish. The main idea of this approach is to add a process for mapping the considered subvectors into the closest codeword in a linear lattice code.

Fix a *q*-ary linear code with the following parameters $[N_i, b]$ in the *i*-th step, where N_i represents the length of code, *b* represents the dimension. Let \mathbf{a}_I represent the subvector consisting of entries corresponding to a collision index set *I*. Rewrite $\mathbf{a}_I = \mathbf{c}_I + \mathbf{e}_I$, where $\mathbf{c}_I \in C_i$ denotes the codeword part and $\mathbf{e}_I \in \mathbb{Z}_q^{N_i}$ denotes the error part. Because of this, the inner product $\langle \mathbf{s}_I, \mathbf{a}_I \rangle$ is equivalent to $\langle \mathbf{s}_I, \mathbf{c}_I \rangle + \langle \mathbf{s}_I, \mathbf{e}_I \rangle$.

Each vector \mathbf{a}_I is sorted by the codeword to which it was mapped. Merging two vectors that are mapped to the same codeword can eliminate $\langle \mathbf{s}_I, \mathbf{c}_I \rangle$ but leave an additional noise term $\langle \mathbf{s}_I, \mathbf{e}_I \rangle$, called coding noise. In order to keep \mathbf{e}_I as small as possible, choosing a suitable decoding process to determine the nearest codeword is a good idea. The noise can be maintained small enough not to affect the whole noise too much by employing a series of lattice codes with various rates.

A full description of the Coded-BKW algorithm consists of five steps. The first three steps are the sample reduction stage, and the last two steps are the hypothesis testing stage. Details are described in Algorithm 1.

Algorithm 1: The Coded-BKW Algorithm
[34]
Input: Matrix $A \in \mathbb{Z}_q^{n imes m}$, received vector
$\mathbf{z} \in \mathbb{Z}_q^m,$ parameters $t_1, t_2, b, n_{test}, l, d.$
begin
Gaussian elimination to change the secret
distribution
Perform t_1 plain BKW steps to eliminate $t_1 \cdot b$
components
Perform t_2 Coded-BKW steps to eliminate
n_{cod} components
for each guess of n_{top} components do
Combine Subspace Hypothesis Testing and
FFT using linear code $[n_{test}, l]$
return reasonable hypothesis

Applying t_1 pure BKW steps aims to balance the merging noise and the coding noise. If not, performing Coded-BKW steps directly will cause the accumulation of coding noise at the beginning of the iteration.

In the hypothesis testing stage, this method applies partial guessing to balance the complexity of the earlier and subsequent procedures. The last step is performed by using subspace hypothesis testing for each guess in the previous part, which we will introduce in "Improvements of hypothesis testing in BKW algorithms" section.

The complexity of Coded-BKW

Let n, q, σ be LWE parameters, $t_1, t_2, b, d, l, n_{test}$ be the parameters in Algorithm 1. Let $n_{cod} = \sum_{i=1}^{t_2} N_i$ be the whole number of components eliminated by t_2 Coded-BKW steps, n_{top} be the remaining unknown positions in the secret $\hat{\mathbf{s}}$ vector. Then let P(d) denote the probability that $|\hat{s}_i| < d$, $\hat{\mathbf{s}}_{test}$ denote the subvector to be tested, P_{test} represent the probability that $||\hat{\mathbf{s}}_{test}|| < \beta \sqrt{n_{tot}}\sigma$ where $n_{tot} = n_{cod} + n_{test}$, β is usually set to be 1.2.

Lemma 2 (Guo et al. 2015) Following the above parameters, the complexity required for a successful run of Coded-BKW for solving LWE is

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

where

$$C_0 = (n+1) \cdot (m-n') \cdot \left\lceil \frac{n'}{b-1} \right\rceil$$

is the complexity of Gaussian Elimination, $n' = n - t_1 b$,

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

is the complexity of t_1 standard BKW steps,

$$C_2 = C'_2 + \sum_{i=1}^{t_2} \left(n_{test} + n_{top} + \sum_{j=1}^{i} N_j \right) \left(M + \frac{(q^b - 1)(i - 1)}{2} \right)$$

is the complexity of t_2 Coded-BKW steps, where $C'_2 = \sum_{i=1}^{t_2} 4N_i \left(\frac{i(q^b-1)}{2} + M\right)$ is the decoding cost, M is the amount of samples following the last Coded-BKW step,

$$C_3 = Mn_{top}(2d+1)^{n_{top}}$$

is the complexity of the partial guessing in the fourth step,

$$C_4 = 4Mn_{test} + (2d+1)^{n_{top}} \left(q^{l+1}(l+1)\log_2 q + q^{l+1} \right)$$

is the complexity of the subspace hypothesis testing in the last step.

The estimated amount of samples needed for testing is

$$M = \frac{4\ln\left((2d+1)^{n_{top}} q^l\right)}{\Delta\left(\chi_{\sigma_{final}} \mid \mid U\right)}$$

where U is the uniform distribution over \mathbb{Z}_q and $\sigma_{final}^2 = 2^{t_1+t_2}\sigma^2 + \beta^2\sigma^2\sigma_{set}^2n_{tot}$, $\Delta(\chi_{\sigma_{final}} ||U)$ is the divergence between two distributions, σ_{set}^2 is a preset variance that is decided by the coding noise utilized in the last phase.

The similarity between Coded-BKW and LMS-BKW is that each iteration does not exactly reduce the vectors to zero but adds some noise. Although the Coded-BKW step can eliminate more components in the treated vectors than the Plain BKW step, it comes with the penalty of introducing an additional noise component, which influences the number of reduction steps. But the coding noise can be maintained small enough that it will not significantly affect the final noise with proper parameter selection. By gradually increasing the step size, Coded-BKW solves the imbalanced final noise distribution of LMS-BKW. The comparison of solving LWE with Coded-BKW for various parameter settings, including Regev's and Lindner-Peikert's cryptosystems, can be found in Guo et al. (2015). Compared to other previous methods, Coded-BKW exhibits a significant performance improvement for all instantiations studied.

The Sieve-Coded-BKW algorithm

In addition to modular switching and linear lattice codes, BKW algorithms also utilize other technologies, such as the use of lattice sieving in Coded-BKW algorithm. Using a sieving step can solve the problem of the growing coding noise in Coded-BKW. Since coding noise will get larger with each step, it must be maintained extremely small in the beginning steps. Using sieving can ensure that the noise from reduced components does not increase but keeps the same size.

Before explaining this new algorithm, we reformulate the LWE problem and the reduction stage in a new manner. Given LWE samples of the form $\mathbf{z} = \mathbf{s}\mathbf{A} + \mathbf{e}$, rewrite this equation as $(\mathbf{s}, \mathbf{e}) \begin{pmatrix} \mathbf{A} \\ \mathbf{I} \end{pmatrix} = \mathbf{z}$, where $\mathbf{M}_0 = \begin{pmatrix} \mathbf{A} \\ \mathbf{I} \end{pmatrix}$.

Decrease the size of columns in \mathbf{M}_i by multiplying the above equation with some particular matrices \mathbf{X}_i . Firstly, multiplying the equation $(\mathbf{s}, \mathbf{e})\mathbf{M}_0 = \mathbf{z}$ by a matrix \mathbf{X}_0 can get $(\mathbf{s}, \mathbf{e})\mathbf{M}_1 = \mathbf{z}_1$, where $\mathbf{M}_1 = \mathbf{M}_0\mathbf{X}_0$, $\mathbf{z}_1 = \mathbf{z}\mathbf{X}_0$. After *t* steps, we have $(\mathbf{s}, \mathbf{e})\mathbf{M}_t = \mathbf{z}_t$, where $\mathbf{M}_t = \mathbf{M}_0\mathbf{X}_0\cdots\mathbf{X}_{t-1}$, $\mathbf{z}_t = \mathbf{z}\mathbf{X}_0\cdots\mathbf{X}_{t-1}$.

The basic ideas of Plain BKW, LMS-BKW, and Coded-BKW can be explained using the procedure described above.

• The reduction procedure of Plain BKW subsequently eliminates rows in matrices \mathbf{M}_i in this way, such that $\mathbf{M}_t = \begin{pmatrix} 0 \\ \mathbf{M}'_t \end{pmatrix}$.

There are only two nonzero items from $\{-1, 1\}$ in each column of X_i . This procedure aims to minimize the size of the entries of columns in M_t .

LMS-BKW and Coded-BKW reduce M_t in a similar way to Plain BKW, but the top rows above M'_t do not need to be eliminated to 0. Rather, positions above M'_t are set to be of the same size as those in M'_t.

Using the above notions and procedures, the sample reduction process of Sieve-Coded-BKW, proposed by Guo et al. (2017), can be introduced in the following.

Assume that we have certain subvectors whose lengths are n_1, n_2, \ldots, n_t respectively and they are concatenated at the first *n* components in columns of **M** (or *n* components in columns of **A**). So $n = \sum_{i=1}^{t} n_i$, $N_j = \sum_{i=1}^{j} n_i, j = 1, \ldots, t$. Let the average size of a reduced component be a constant *B*. The goal of Sieve-Coded-BKW is to make the average size of vectors whose length is n' less than $\sqrt{n'} \cdot B$. The *i*-th iterative reduction process can be divided into two steps, as shown in Fig. 2.

The first step CodeMap(\mathbf{m}, i): For all columns $\mathbf{m} \in \mathbf{M}_{i-1}$, consider the positions from $N_{i-1} + 1$ to N_i . Map the subvector whose length is n_i to the closest codeword in C_i of the same length, the distance should be smaller than $\sqrt{n_i} \cdot B$. The closest codeword



Fig. 2 The core ideas of Sieve-Coded-BKW in the *i*th reduction step

is the output of CodeMap(\mathbf{m} , i), and store \mathbf{m} in the list \mathcal{L}_{Δ} .

The second step Sieve(L_Δ, *i*, √N_i · B): Compute all differences between any pairs of vectors in the list L_Δ. Considering the first N_i positions, if the norm of the difference is smaller than √N_i · B, put it in the list S_Δ. Then store all S_Δ in M_i.

After repeating the reduction *t* times, the average norm of the first *n* components of columns in $\mathbf{M}_{\mathbf{t}}$ is smaller than $\sqrt{n} \cdot B$. The resulting samples roughly follow Gaussian distribution $\chi_{\sigma^2 \cdot (nB^2+2^t)}$, then one can use a distinguisher to confirm whether a hypothesis about secret values is correct or not. The entire process can be repeated after obtaining some correct secret values with a smaller dimension.

The concrete instantiation of the Sieve-Coded-BKW algorithm is shown in Algorithm 2.

Algorithm 2: Concrete instantiation of Sieve-Coded-BKW [35]
Input: Matrix $A \in \mathbb{Z}_q^{n imes m}$, vector $\mathbf{z} \in \mathbb{Z}_q^m$,
parameters $t_0, t_{1a}, t_{1b}, B, n_i (1 \leq i \leq t).$
begin
Change the secret distribution by Gaussian
elimination
Try t_0 Plain BKW steps
Try t_{1a} Coded-BKW steps
Try t_{1b} Sieve-Coded-BKW steps
Subspace hypothesis testing with FFT

Using the t_0 Plain BKW reduction steps for pre-processing can avoid the massive accumulation of coding noise at the beginning of the algorithm. Using the t_{1a} Coded-BKW steps will be more effective than always using sieving when the dimension is relatively small. This algorithm significantly outperforms the previous best BKW variants for some specific parameters, that will be demonstrated in "Comparisons" section.

Improvements of Sieve-Coded-BKW

Assume that after i - 1 reduction steps, the average size of the first N_{i-1} components of the vector **a** is less than the constant B_{i-1} . After *i* reduction steps, the average size of the first N_i components is less than the constant B_i , satisfying $B_i = \gamma B_{i-1}$.

The Sieve-Coded-BKW algorithm is a particular case when $\gamma = 1$. Because using sieving can ensure that the noise from reduced components does not increase but keeps the same size.

Guo et al. (2019) introduced a new variant of Sieve-Coded-BKW by applying the nearest neighbor searching after Coded-BKW, that was called the γ -Sieve-Coded-BKW algorithm. This algorithm selects the same reduction factor γ at each iteration step, where γ is not limited to 1 but satisfies $0 < \gamma \leq \sqrt{2}$.

Mårtensson (2019) continued to improve the γ -Sieve-Coded-BKW algorithm by increasing the reduction factors γ_i in different sieving steps, and the improved new algorithm is called γ_i -Sieve-Coded-BKW. In the beginning steps, sieving is so cheap that a small γ_i can be employed. With sieving getting more and more costly, it's better to increase γ_i . Assume that there are t_2 reduction steps of γ_i -Sieve-Coded-BKW in total. Let $\gamma_1 = \gamma_s$ and $\gamma_{t_2} = \gamma_f$. The authors let

$$\gamma_i = \gamma_s + \frac{\gamma_f - \gamma_s}{t_2 - 1}(i - 1).$$

The comparison of asymptotic complexity between three types of the Coded-BKW with sieving algorithms can be seen in Table 1 of this paper (Mårtensson 2019). The asymptotic complexity of the γ_i -Sieve-Coded-BKW algorithm performs best for all parameter settings.

The BKW-FWHT-SR algorithm

In order to reduce more positions in a finite number of iterations, a modified reduction step of lazy modulus switching was presented by Budroni et al. (2021), which was called Smooth-LMS. This method employs easy LMS to partially reduce one additional position after reducing a given number of positions. The new algorithm also improves the hypothesis testing stage to recover the secret by mapping LWE to a binary problem and utilizing the FWHT distinguisher, as discussed in "Improvements of hypothesis testing in BKW algorithms" section. The whole algorithm was named BKW-FWHT-SR.

Given some samples of the form $\mathbf{z} = \mathbf{sA} + \mathbf{e} \mod q$, where **A** has at most 2^{ν} columns. This method can reduce the size of some components to less than B_i , $i = 1, \dots, t$. In other words, the size of these components will be in the set $\{-B_i + 1, \dots, 0, 1, \dots, B_i - 1\}$.

In the first reduction step, check all columns in **A**. Simply write column *i* as $\mathbf{x} = (x_1, \dots, x_n)$ and compute

$$k_{j} = \begin{cases} x_{j} \operatorname{div} B_{1}, x_{1} \ge 0 \\ -x_{j} \operatorname{div} B_{1}, x_{1} < 0 \end{cases}, \quad j = 1, \dots, n_{1}$$
$$k_{n_{1}+1} = \begin{cases} x_{n_{1}+1} \operatorname{div} B_{1}^{'}, x_{1} \ge 0 \\ -x_{n_{1}+1} \operatorname{div} B_{1}^{'}, x_{1} < 0 \end{cases},$$

where n_1 is the number of fully reduced positions, $n_1 + 1$ is the partially reduced position in the first reduction.

Put these vectors $K_i = (k_1, k_2, ..., k_{n_1+1})$ in a sorted list \mathcal{L} . If the integers divided by B_1 have the same values, the samples that are reduced will be in the same category

After inserting each column into the list \mathcal{L} , a new matrix **A** can be built. Check all columns and merge each pair of columns in the same category by adding or subtracting to create new columns in **A**. When there are 2^{ν} new columns in total, stop this process. The size of positions $\{1, \ldots, n_1\}$ and position $n_1 + 1$ in the new matrix **A** are less than B_1 and less than B'_1 , respectively.

In the next *l* iterations, continue to check each column in **A**. Simply write column *i* as $\mathbf{x} = (x_1, \ldots, x_n)$. For the $\{N_{l-1} + 1, \cdots, N_l + 1\}$ positions, calculate

$$k_{j} = \begin{cases} x_{N_{l-1} + j} \operatorname{div} B_{l}, x_{N_{l-1} + 1} \ge 0 \\ -x_{N_{l-1} + j} \operatorname{div} B_{l}, x_{N_{l-1} + 1} < 0 \end{cases}, \quad j = 1, \dots, n_{l}$$
$$k_{n_{l}+1} = \begin{cases} x_{N_{l}+1} \operatorname{div} B'_{l}, x_{N_{l-1} + 1} \ge 0 \\ -x_{N_{l}+1} \operatorname{div} B'_{l}, x_{N_{l-1} + 1} < 0 \end{cases}.$$

Put these vectors $K_i = (k_1, k_2, ..., k_{n_l+1})$ in a sorted list \mathcal{L} . After inserting all columns into the list, a new matrix **A** can be built as in the first step.

After *t* iterations, the resulting samples will have the form $z_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i$, where the norm of each \mathbf{a}_i has been reduced. The term on the right-hand side may be roughly represented as a discrete Gaussian sample, if the norm of the columns in **A** is small enough. Furthermore, we can distinguish the samples z_i with samples from a uniform distribution, if the standard deviation is small.

We can also apply Smooth-LMS to Plain BKW, which is called Smooth-plain BKW steps. In the first step, set $B_1 = 1$ and $B'_1 > 1$, and the reduced vector **x** will have $x_1 = \cdots = x_{n_1} = 0$ and $|x_{n_1+1}| < B'_1$. Employing Smooth-plain BKW steps can reduce some extra positions in the sample reduction.

The sample reduction of the BKW-FWHT-SR algorithm has the identical time and memory complexity in each iteration step, which can balance the whole complexity in this stage and reduce more components in total. When it comes to complexity, this algorithm performs best among all other previous methods. Its complexity theorem for solving LWE can be found in paper (Budroni et al. 2021).

A high-level comparison of the sample reduction stage

A comprehensive comparison between the features of the sample reduction of different BKW algorithms is shown in Figs. 3, 4, 5 and Table 1. Figure 3, a similar version of figure 7.1 from Mårtensson (2020), is a depiction of how the values of the **a** vectors vary during the reduction stage for various BKW algorithms. Figure 4, a similar version of Fig. 2 from Mårtensson



Fig. 4 A comparison of different types of Coded-BKW combined with sieving



Fig. 3 A comprehensive comparison of sample reduction of different BKW algorithms



Fig. 5 A description of how the Smooth-LMS and Smooth-plain BKW perform better than their standard equivalents (Budroni et al. 2021)

Table 1 The connection between B_i and B_{i-1} in sample reduction of BKW algorithms

The BKW algorithms	The connection between B_i and B_{i-1}
Plain BKW	$B_i = B_{i-1} = 0$
LMS-BKW/Coded-BKW	$B_i = \sqrt{2}B_{i-1}$
Sieve-Coded-BKW	$B_i=B_{i-1}\neq 0$
γ-Sieve-Coded-BKW	$B_i = \gamma B_{i-1}$
γ_i -Sieve-Coded-BKW	$B_i = \gamma_i B_{i-1}$
BKW-FWHT-SR	$B_i = \lfloor \sqrt{2}B_{i-1} \rceil$

(2019), is a depiction of various Coded-BKW with sieving. Figure 5, a similar version of Fig. 1 from Budroni et al. (2021), shows how the Smooth-LMS and Smoothplain BKW perform better than their standard equivalents by partially reducing an additional component in each iteration. In these figures, the horizontal axis denotes components in vectors **a**, the vertical axis denotes the mean norm of the relevant component. The yellow color represents reduced positions, and the purple color represents unreduced positions.

The sample reduction of each BKW algorithm has its own unique characteristics, but they can also be viewed within a unified framework. Assume that after i - 1 reduction steps, the first N_{i-1} positions have been reduced to B_{i-1} . After *i* steps, the first N_i positions of new sample vectors have been reduced to B_i . Table 1 gives a generic BKW reduction framework about the changes of absolute values in the **a** vectors. From the above figures and table, we draw the following conclusions about the sample reduction of various BKW algorithms.

- Plain BKW aims to get zero vectors by adding or subtracting samples in each iteration such that *b* components of columns in **A** reduced to 0, which is equivalent to setting $B_i = B_{i-1} = 0$.
- LMS-BKW/Coded-BKW reduces components of **a** to a small value, but not to zero, which is equivalent to setting $B_i = \sqrt{2B_{i-1}}$.
- LMS-BKW maps samples to the same group if the reduced components yield the same value when divided by an appropriate modulus *p*. However, the distribution of the **a** vectors becomes uneven as a result of the size of the earlier reduced components increasing step by step.
- Coded-BKW maps samples to the same group if the codeword they are mapped to is identical. The distribution of vectors **a** is even as a result of the step sizes steadily increasing and the degree of reduction gradually decreasing.
- Sieve-Coded-BKW ensures that the earlier reduced components do not increase, which is equivalent to setting $B_i = B_{i-1} \neq 0$ for $\gamma = 1$. In contrast to Coded-BKW, Sieve-Coded-BKW does not initially need to reduce the components as much. However, the more positions we work with, the more expensive

the sieving process gets. Consequently, the step size must be gradually reduced. The improved version of γ -Sieve-Coded-BKW sets $B_i = \gamma B_{i-1}$ for an invariable γ . γ_i -Sieve-Coded-BKW sets $B_i = \gamma_i B_{i-1}$ for different values of γ_i in different reduction steps.

• BKW-FWHT-SR, which supports non-integer step sizes, utilizes Smooth-LMS to entirely eliminate a given number of components and partly reduce an additional position. This method can balance the whole complexity and reduce more components in total. In each iteration *i*, it reduces some positions to be less than B_i , for $i = 1, \dots, t$. The connection between the magnitudes of the components is related to the constant B_i .

Improvements of hypothesis testing in BKW algorithms

After sample reduction, there are k positions of the secret vector **s** for the hypothesis testing stage, which are denoted as $\mathbf{s} = (s_1, \ldots, s_k)$. The challenge is to identify the correct guess among all the other q^k guesses. The observed error values are uniformly random for the incorrect guess, whereas they follow a Gaussian distribution for the correct one. We will introduce the distinguishing methods for BKW algorithms in this section.

In the following parts, we denote the guesses of the secret sub-vector as \hat{s} . The errors $e_i = z_i - \sum_{j=1}^k a_{i_j} \cdot \hat{s}_j$, i = 1, 2, ..., m follow a Gaussian distribution with standard deviation σ_{final} and mean 0.

Optimal distinguisher

The optimal distinguisher for the hypothesis testing stage is an exhaustive search method. It aims to distinguish the hypothesis $D_{\hat{\mathbf{s}}} = U$ against $D_{\hat{\mathbf{s}}} = \chi_{\sigma_{final}}$, *q*, where *U* is the uniform distribution over \mathbb{Z}_q and $\chi_{\sigma_{final}}$, *q* is the error distribution when the guess is $\hat{\mathbf{s}}$.

Compute the log-likelihood ratio

FFT distinguisher

The FFT-BKW (Duc et al. 2015) is the first BKW algorithm that uses the FFT technique in the hypothesis testing stage. It slightly changes the sample reduction of Plain BKW by removing the final iteration. As a result, the samples used for the hypothesis testing would have noise made up of 2^{a-1} Gaussians rather than 2^{a} .

Now we focus on the FFT distinguisher in the hypothesis testing stage. Given *m* samples (\mathbf{a}_i, z_i) after the sample reduction, where every \mathbf{a}_i has all of its elements equal to zero with the exception of a block that has a size k = n - b(a - 1). Represent the *m* samples as a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times k}$ and a vector $\mathbf{z} \in \mathbb{Z}_q^m$, with \mathbf{A}_j as the rows.

Remember that $\theta_q := e^{2\pi i/q}$, write the following function

$$f(\mathbf{x}) := \sum_{j=1}^m \mathbb{I}_{\{\mathbf{A}_j = \mathbf{x}\}} \theta_q^{\mathbf{z}_j}, \forall \mathbf{x} \in \mathbb{Z}_q^k.$$

If $\pi(x)$ is true, then $\mathbb{I}_{\{\pi(x)\}} = 1$; otherwise, it is zero. The FFT distinguisher works by computing the FFT of *f*, that is,

$$\hat{f}(\boldsymbol{\alpha}) := \sum_{\mathbf{x} \in \mathbb{Z}_q^k} f(\mathbf{x}) \theta_q^{-\langle \mathbf{x}, \boldsymbol{\alpha} \rangle} = \sum_{j=1}^m \theta_q^{-(\langle \mathbf{A}_j, \boldsymbol{\alpha} \rangle - \mathbf{z}_j)}.$$

When there are sufficient samples relative to the noise level, then the right guess $\boldsymbol{\alpha} = \mathbf{s}$ results in the maximum value of $\operatorname{Re}(\hat{f}(\boldsymbol{\alpha}))$.

The FFT distinguisher costs $\mathcal{O}(m + k \cdot q^k \cdot \log q)$, which is much lower than $\mathcal{O}(m \cdot q^k)$ in general. The upper bound of sample complexity of the FFT distinguisher is $8 \cdot \ln\left(\frac{q^k}{\varepsilon}\right) \cdot \left(1 - 2\pi^2\sigma^2/q^2\right)^{-2^{t+1}}$ according to Duc et al. (2015), where ε is the probability of guessing **s** incorrectly. The whole complexity to solve Search-LWE with FFT-BKW is given in Theorem 17 of Duc et al. (2015).

$$\sum_{e} F(e) \log \frac{\Pr_{\chi_{\sigma_{final} \dots, q}}(e)}{\Pr_{U}(e)} = \sum_{e} F(e) \log \left(q \cdot \Pr_{\chi_{\sigma_{final} \dots, q}}(e) \right),$$

where F(e) represents the frequency of occurrence of e when guessing $\hat{\mathbf{s}}$, $\Pr_D(e)$ represents the probability of choosing e from the distribution D. Select $\hat{\mathbf{s}}$ that optimizes the equation by maximizing its value.

The computational time needed by this method is $\mathcal{O}(m \cdot q^k)$ if all potential hypotheses are tested. With the secret-noise transformation applied, the complexity can drop to $\mathcal{O}(m \cdot (2d+1)^k)$ as the absolute values of k components in **s** will be less than d.

Pruned FFT distinguisher

By limiting the number of hypotheses, Guo et al. (2021) introduced a new method called the Pruned FFT distinguisher. In general, only a few output values of the FFT distinguisher are needed, so limiting the size of the k components of **s** to d is a good idea.

This method changes the complexity of distinguishing into $\mathcal{O}(m + k \cdot q^k \cdot \log (2d + 1))$. Furthermore, the

maximum limit of sample complexity is also reduced to $8 \cdot \ln\left(\frac{(2d+1)^k}{\varepsilon}\right) \cdot \left(1 - 2\pi^2 \sigma^2/q^2\right)^{-2^{t+1}}$.

Subspace hypothesis testing

The subspace hypothesis testing was originally presented for LPN in the \mathbb{Z}_2 case, then generalized to the \mathbb{Z}_q case to efficiently calculate the frequencies of different symbols in \mathbb{Z}_q (Guo et al. 2015).

Utilize an $[n_{test}, l]$ systematic linear code to sort the samples $(\hat{\mathbf{a}}_i, \hat{z}_i)$ from sample reduction, which was described in "The Coded-BKW algorithm" section by their nearest codewords \mathbf{c}_i . Write the function

$$f^{\mathbf{c}_i}(X) = \sum_{(\mathbf{a}_i, z_i)} X^{\hat{z}_i \pmod{q}}.$$

Write $h_{\mathbf{u}}(X) = f^{\mathbf{c}_i}(X)$, where **u** is the information part of the codeword and has q^l possible values. Define

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

Among all candidates of $\mathbf{y} \in \mathbb{Z}_q^l$, there exists only one unique $\mathbf{y} \in \mathbb{Z}_q^l$ that satisfies $\langle \mathbf{y}, \mathbf{u} \rangle = \langle \hat{\mathbf{s}}, \mathbf{c}_i \rangle$. The polynomial $H_{\mathbf{y}}(X)$, which can be sped up by FFT, will record the frequency of the Gaussian error symbols if the guess is correct; otherwise, it will be uniformly randomly distributed.

Following the notations from "The Coded-BKW algorithm" section, the overall complexity of subspace hypothesis testing is

$$\mathcal{O}\Big(M \cdot n_{test} + (2d+1)^{n_{top}} \cdot \Big(q^{l+1} \cdot \log_2 q^{l+1} + q^{l+1}\Big)\Big).$$

FWHT distinguisher

The FWHT distinguisher was also originally proposed for LPN in the \mathbb{Z}_2 case (Chose et al. 2002).

Let *k* represent a *n*-bit vector as $(k_0, k_1, \ldots, k_{n-1})$ and consider a sequence X_k that denotes the number of occurrences of X = k, $k = 0, 1, \ldots, 2^n - 1$. Given samples in the format $\mathbf{z} = \mathbf{sA} + \mathbf{e} \mod 2$, write

$$X_k = \sum_{j \in L} (-1)^{z_j},$$

where *L* is the set of all columns in **A** that match the constant *k*.

The Walsh-Hadamard transform is defined as

$$\hat{X}_w = \sum_{k=0}^{N-1} X_k \cdot (-1)^{w \cdot k},$$

where $w \cdot k$ represents the multiplication of the *n*-bit indices *w* and *k*. Compute $|\hat{X}_{\bar{w}}| = \max_{w} |\hat{X}_{w}|$, which corresponds to the correct **s**. The FWHT technique can accelerate WHT with time complexity $\mathcal{O}(N \log N)$.

The BKW-FWHT-SR algorithm (Budroni et al. 2021) mapped the LWE problem to a binary LPN problem and applied the fast Walsh Hadamard transform, which can quickly and accurately guess a large number of entries. This kind of method can precisely distinguish the secret at a higher noise level and perform better than the FFTbased method. Additionally, the FWHT is implemented considerably more quickly.

A comparison of various distinguishers in the hypothesis testing stage

A detailed comparison of the characteristics of various distinguishers as well as the corresponding time and sample complexity is shown in Table 2. The parameters that appear in the formula in Table 2 can be found in the corresponding algorithms in "Improvements of sample reduction in BKW algorithms".

Comparisons

This section begins with a unified framework for the BKW algorithms, followed by a comparison of the characteristics of each BKW algorithm in Table 3. Then we estimate the concrete security of some specific LWE instances by using all BKW algorithms and also compared them with lattice-based algorithms. The results are given in Tables 4 and 5.

 Table 2
 A
 comparison
 of
 different
 distinguishers
 in
 the

 hypothesis testing stage

Distinguisher	Time complexity
	Sample complexity
Optimal distin- guisher	$\mathcal{O}\left(m\cdot(2d+1)^k\right)$
	$8 \cdot \ln\left(\frac{q^k}{\varepsilon}\right) \cdot \left(1 - 2\pi^2 \sigma^2/q^2\right)^{-2^{l+1}}$
FFT distinguisher	$\mathcal{O}(m+k\cdot q^k\cdot \log q)$
	$8 \cdot \ln\left(\frac{q^k}{\varepsilon}\right) \cdot \left(1 - 2\pi^2 \sigma^2/q^2\right)^{-2^{t+1}}$
Pruned FFT distin-	$\mathcal{O}\left(m+k\cdot q^k\cdot \log\left(2d+1\right)\right)$
guisher	$8 \cdot \ln\left(\frac{(2d+1)^{-k}}{\varepsilon}\right) \cdot \left(1 - 2\pi^2 \sigma^2/q^2\right)^{-2^{t+1}}$
Subspace hypoth-	$\mathcal{O}\left(M \cdot n_{test} + (2d+1)^{n_{top}} \cdot \left(q^{l+1} \cdot \log_2 q^{l+1} + q^{l+1}\right)\right)$
esis testing	$\frac{4\ln\left((2d+1) {}^{n} \ top \ q^{l}\right)}{\Delta\left(\chi_{\sigma_{final}} \parallel U\right)}$
FWHT distin-	$\mathcal{O}(2^k \log 2^k)$
guisher	$4\ln(2^{n}t)$
	$\Delta(\chi\sigma_{final}, 2q \parallel U 2q)$

BKW algorithms	Sample reduction	Hypothesis testing
Plain BKW	Reduces a fixed number of positions to zero	Optimal distinguisher
FFT-BKW	Same as Plain BKW except removing the final iteration	FFT distinguisher
LMS-BKW	Reduces a fixed number of positions to a small value not to zero by combining lazy modulus switching	Optimal distinguisher
Coded-BKW	Reduces an increasing number of positions to a small value not to zero by combining linear lattice codes	Subspace hypothesis testing + FFT
Sieve-Coded-BKW	Reduces a decreasing number of positions to a small value not to zero by combining linear lattice codes and sieving. Different cases with different reduction factors	Subspace hypothesis testing + FFT
BKW-FWHT-SR	Fully reduces a given amount of positions and partially reduces an additional position to configured values	Map LWE to LPN + FWHT distinguisher

 Table 3
 A comparison of BKW algorithms in sample reduction and hypothesis testing

Table 4 Concrete security estimation of LWE instances in the TU Darmstadt LWE challenge

Parameters		The BKW algorithms							The lattice-based algorithms		
n	9	α	Plain BKW	LMS-BKW	FFT-BKW	Coded-BKW	Sieve- Coded- BKW	BKW-FWHT-SR	Primal	Decoding	Dual
40	1601	0.005	47.3	47.4	40.6	42.6	41.5	34.4	31.4	35.0	37.1
		0.010	51.6	49.9	43.5	43.7	42.7	39.3	31.7	35.0	37.6
		0.015	55.2	52.2	46.5	52.6	44.1	42.4	32.0	35.2	41.4
		0.020	57.4	53.4	49.0	52.6	49.1	46.2	32.4	35.3	43.8
		0.025	59.9	54.8	52.6	52.7	49.2	48.3	34.3	36.6	48.1
		0.030	62.5	56.6	52.6	52.7	50.4	50.0	37.9	39.0	53.4
45	2027	0.005	51.7	51.6	44.1	55.2	45.0	37.7	31.6	35.1	37.4
		0.010	56.6	54.4	48.8	55.2	45.3	43.5	32.0	35.3	39.9
		0.015	60.8	57.0	51.8	55.2	54.7	48.3	32.5	35.5	43.9
		0.020	63.3	58.5	55.4	55.2	54.8	51.2	35.7	37.3	50.7
		0.025	66.1	60.2	55.7	55.3	54.8	54.1	39.9	41.0	57.1
		0.030	72.8	65.2	59.7	64.1	63.3	56.3	44.2	45.1	64.1
50	2503	0.005	56.3	55.7	48.9	46.4	45.5	41.8	31.9	35.2	37.6
		0.010	61.6	59.0	54.4	56.0	53.3	48.7	32.5	35.5	42.5
		0.015	66.3	61.9	57.8	56.8	53.4	52.5	35.5	37.2	50.5
		0.020	72.2	66.2	61.9	61.9	60.4	56.4	41.0	42.7	59.0
		0.025	72.2	65.3	61.9	66.1	61.7	59.3	46.4	47.7	65.4
		0.030	75.7	67.7	66.9	66.3	65.6	63.3	51.4	52.9	75.5
70	4903	0.005	75.9	74.0	66.2	62.3	62.2	62.3	34.6	37.5	49.5
		0.010	82.8	79.5	72.9	67.1	70.6	73.7	47.1	49.6	66.3
		0.015	88.2	81.1	77.5	73.3	72.7	75.6	57.4	60.9	81.2
120	14401	0.005	125.2	119.1	113.4	110.5	108.8	100.1	70.3	79.4	98.4
		0.010	142.0	130.4	123.1	124.0	115.8	115.1	86.5	101.7	106.1
		0.015	148.9	133.8	130.9	136.8	135.3	127.0	98.3	118.5	133.3

Among these different estimation methods, bold-faced numbers are the smallest

A general framework and characteristics of BKW algorithms

shown in Algorithm 3.

All various BKW algorithms can be described by a generic framework when solving the LWE problem, as

Here are m LWE samples. If more samples are required, we first apply sample amplification. Additionally, if the secret's standard deviation is larger than that

Parameters		The BKW Algorithms							The Lattice-based Algorithms		
n	q	σ	Plain BKW	LMS-BKW	FFT-BKW	Coded-BKW	Sieve- Coded- BKW	BKW-FWHT-SR	Primal	Decoding	Dual
Regev	, parameter	s									
128	16411	11.81	119.4	114.7	107.5	84.5	84.2	59.2	57.3	61.9	69.2
256	65537	25.53	269.6	220.3	200.8	145.1	130.0	107.0	103.6	121.7	121.0
512	262147	57.06	429.4	403.4	384.6	287.6	247.6	243.3	201.6	252.7	231.2
Lindn	er-Peikert po	arameters									
128	2053	2.7	104.4	100.0	95.7	69.7	69.2	48.8	53.4	57.1	67.5
256	4099	3.34	181.8	176.0	167.9	123.8	112.9	98.5	95.2	111.8	112.3
512	4099	2.9	338.8	327.7	308.0	209.2	197.3	188.7	179.0	226.5	207.8

Table 5 Concrete security estimation of LWE instances from Regev's and Lindner-Peikert's cryptosystems

Among these different estimation methods, bold-faced numbers are the smallest

of the error, we apply the secret-error transform before reduction. Then in the sample reduction stage, we perform some Plain BKW steps, followed by other new reduction steps. The purpose of some Plain BKW steps is to avoid the accumulation of a large amount of noise at the beginning of the algorithm. After that, we choose a suitable distinguisher to guess some positions of the secret vector. In the end, we backtrack to the previous step if needed.

Algorithm 3: A General Framework of BKW algorithms
Input: Matrix $A \in \mathbb{Z}_q^{n imes m}$, received vector
$\mathbf{z} \in \mathbb{Z}_q^m,$ parameters $t_1, t_2.$
begin
Change the secret distribution by Gaussian
elimination
Try t_1 Plain BKW steps
Try t_2 other BKW reduction steps
Use suitable distinguisher to guess some
positions

Within the same framework, each BKW algorithm has its characteristics, which we summarize in Table 3 by presenting the features of their sample reduction and hypothesis testing stages.

Concrete security estimation

In the end, by using the BKW algorithms and also three lattice attacks (the primal attack, decoding attack, and dual attack) (Albrecht et al. 2015a), we estimate the concrete security of LWE instances from TU Darmstadt LWE challenge (https://www.latticechallenge.org/lwe_ challenge/challenge) and Regev's (2005)/Lindner–Peikert's (2011) cryptosystems to present the comparisons among them. The estimation results are given in Tables 4 and 5. Among these different methods of estimation, bold-faced numbers are the smallest.

In "Improvements of sample reduction in BKW algorithms", we studied the complexity of Plain BKW and Coded-BKW for given parameters n, q, σ in Lemmas 1 and 2. The complexity of LMS-BKW, FFT-BKW, Sieve-Coded-BKW, and BKW-FWHT-SR can be found in Albrecht et al. (2014), Duc et al. (2015), Guo et al. (2017), Budroni et al. (2021), respectively. According to these, we can compute the overall complexity of these BKW algorithms by building some Sage (http://www. sagemath.org/doc/tutorial/) modules with the latticeestimator (https://github.com/malb/lattice-estimator). The results of the lattice attacks in the last three columns invoked the latest update of the lattice-estimator (https://github.com/malb/lattice-estimator). Both the sieving and enumeration cost models are used, but only the lower values are selected.

To simplify our complexity calculation, we make the following assumptions.

- The complexity of operations over C and Z_q is equal in our estimation.
- There are infinite samples.
- Take $C_{FFT} = 1$, and the successful probability $\varepsilon = 0.99$.

In the calculation of Table 4, for the sake of comparison, we select the parameters n, q, α according to Table 2 in Budroni et al. (2021). For instance, we estimate the complexities by implementing estimator codes for $n = 40, q = 1601, \alpha = 0.005$. Plain BKW takes $2^{47.3}$ operations, while BKW-FWHT-SR requires $2^{34.4}$ operations. The best result of the lattice attacks needs $2^{31.4}$ operations according to the LWE-estimator. In calculating Table 5, we choose n = 128, 256, 512. The cost values of these algorithms can be computed in a similar way using Sage and the LWE-estimator.

Discussion. From Tables 4 and 5, we can see that the BKW-FWHT-SR algorithm significantly outperforms other BKW algorithms in most cases, while Primal attack has the lowest cost among lattice attacks in all cases. For fixed n and q, the costs of all algorithms increase as noise increases.

In general, Primal attack beats all kinds of BKW algorithms for almost all values of *n*, except for n = 128, q = 2053. In this parameter setting, the BKW-FWHT-SR algorithm appears to be more efficient.

In conclusion, the BKW algorithms, which use different techniques, have been improved to some extent compared with Plain BKW. However, compared with lattice attacks, the BKW algorithms are less efficient in practice. In addition, compared with lattice attacks, the sample complexity needed in BKW algorithms is much greater. For example, under the Regev/Lindner-Peikert parameters, the lattice-based algorithms only require about hundreds of samples in our implementation, while the BKW algorithms require exponential samples. In practice, we can not get as many samples as they need, which is the biggest limitation of BKW algorithms.

Conclusion

In this paper, we provide a review of the evolution of BKW algorithms for solving LWE over the last decade. We describe the various methods used for the sample reduction stage and the hypothesis testing stage, respectively. We also compare these BKW algorithms by estimating the concrete security of LWE with different parameters, and we also compare them with lattice-based algorithms.

The BKW algorithms have an advantage in concrete security comparisons on LWE with certain parameters. However, there are still many unresolved problems in practical applications, among which the most important one is that it needs an unlimited number of samples and a large memory to store them.

There are still many potential points for innovation about BKW in the future. For sample reduction stages, there are essentially two different possible ways: choosing what steps to take or improving the individual step in an iteration. Accordingly, there are several directions for improvement that can be considered, as follows. First, we consider organizing the reduction stage in a wiser manner. For example, for the Coded-BKW with sieving algorithm, we can decrease the reduction factors of the sample reduction stage and use quantum Grover's algorithm to speed up the iteration. Second, we can reduce the positions to different set magnitudes by using sieving or modulus switching to improve the individual step in an iteration. Third, we can combine the idea of k-BKW for solving LPN (Esser et al. 2018) with the Coded-BKW reduction steps instead of only merging two samples in an iteration. We can also adopt a hybrid method to strike a balance between the complexity of the sample reduction and hypothesis testing stages. In addition, combining other new techniques with BKW to improve its implementation in practice is also a pressing research question.

Acknowledgements

We would like to thank the anonymous reviewers and editors for detailed comments and useful feedback.

Author contributions

WY and BL completed the drafted manuscripts of the paper. LXH and WKP participated in problem discussions. All authors read and approved the final manuscript.

Funding

This work is supported by National Natural Science Foundation of China (No. U1936209).

Availability of data and materials

No applicable.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 23 January 2023 Accepted: 12 April 2023 Published online: 03 September 2023

References

- Abdalla M, Fouque P, Lyubashevsky V, Tibouchi M (2012) Tightly-secure signatures from lossy identification schemes. In: Annual international conference on the theory and applications of cryptographic techniques. Springer, pp 572–590
- Agrawal S, Libert B, Stehle D (2015) Fully secure functional encryption for inner products, from standard assumptions. Cryptology ePrint Archive, Paper 2015/608. https://eprint.iacr.org/2015/608
- Albrecht M, Faugre J (2012) On the complexity of the Arora-Ge algorithm against LWE. In: Proceedings of the 3rd international conference on symbolic computation and cryptography, pp 93–99
- Albrecht M, Fitzpatrick R (2013) On the efficacy of solving LWE by reduction to unique-SVP. In: ICISC 2013, pp 293–310
- Albrecht M, Faugere J, Fitzpatrick R (2014) Lazy modulus switching for the BKW algorithm on LWE. PublicKey Cryptogr PKC 8383:429–445
- Albrecht M, Player R, Scott S (2015a) On the concrete hardness of learning with errors. J Math Cryptol 9:169–203
- Albrecht M, Cid C, Faugère J (2015b) On the complexity of the BKW algorithm on LWE. Des Codes Cryptogr 74:325–354
- Alkim E, Ducas L, Pöppelmann T, Schwabe P (2015) Post-quantum key exchange—a new hope. Cryptology ePrint Archive, Paper 2015/1092. https://eprint.iacr.org/2015/1092
- Applebaum B, Cash D, Peikert C (2009) Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: CRYPTO'09, pp 595–618

Becker A, Ducas L, Gama N, Laarhoven T (2016) 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

- Bi L, Li S, Liu Y, Zhang J, Fan S (2019) A survey on the analysis of the concrete hardness of LWE. J Cyber Secur 4(2):1–12
- Blum A, Kalai A, Wasserman H (2003) Noise-tolerant learning, the parity problem, and the statistical query model. J ACM 50:506–519
- Bogos S, Vaudenay S (2016) Optimization of LPN solving algorithms. In: Advances in cryptology—ASIACRYPT, pp 703–728
- Bogos S, Tramèr F, Vaudenay S (2015) On solving LPN using BKW and variants. In: Cryptography and communications, vol 8, p 3
- Boyen X (2013) Attribute-based functional encryption on lattices. In: Theory of cryptography conference on theory of cryptography
- Brakerski Z, Vaikuntanathan V (2014) Efficient fully homomorphic encryption from (standard) LWE. SIAM J Comput 43(2):831–871
- Brakerski Z, Vaikuntanathan V (2016) Circuit-ABE from LWE: unbounded attributes and semi-adaptive security. Springer, Berlin
- Budroni A, Guo Q, Johansson T (2021) Improvements on making BKW practical for solving LWE. Cryptography 5:31
- Chen Y, Nguyen PQ (2011) Bkz 2.0: better lattice security estimates. In: ASIA-CRYPT 2011, pp 1–20
- Chose P, Joux A, Mitton M (2002) Fast correlation attacks: an algorithmic point of view. In: Advances in cryptology—EUROCRYPT 2002, pp 209–221
- Ding J, Xie X, Lin X (2012) A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive
- Dong X, Hua J, Sun S, Li Z, Wang X, Hu L (2021) Meet-in-the-middle attacks revisited: key-recovery, collision, and preimage attacks. Cryptology ePrint Archive, Paper 2021/427. https://eprint.iacr.org/2021/427
- Duc A, Tramèr F, Vaudenay S (2015) Better algorithms for LWE and LWR. In: Advances in cryptology—EUROCRYPT, pp 173–202
- Esser A, Heuer F, Kübler R (2018) Dissection-BKW. In: Cryptology—CRYPTO 2018, pp 638–666
- Gentry C (2009) A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University
- Goldwasser S, Gordon SD, Goyal V, Jain A, Katz J, Liu F-H, Sahai A, Shi E, Zhou H-S (2014) Multi-input functional encryption. In: Annual international conference on the theory and applications of cryptographic techniques, pp 578–602
- Güneysu T, Lyubashevsky V, Pöppelmann T (2012) Practical lattice-based cryptography: a signature scheme for embedded systems. In: International workshop on cryptographic hardware and embedded systems. Springer, pp 530–547
- Guo Q, Johansson T, Londahl C (2014) Solving LPN using covering codes. In: ASIACRYPT, pp 1–20
- Guo Q, Johansson T, Stankovski P (2015) Coded-bkw: solving LWE using lattice codes. In: Advances in cryptology—CRYPTO, pp 23–42
- Guo Q, Johansson T, Mårtensson E (2017) Coded-BKW with sieving. In: Advances in cryptology—ASIACRYPT, pp 323–346
- Guo Q, Johansson T, Mårtensson E (2019) On the asymptotics of solving the LWE problem using coded-BKW with sieving. IEEE Trans Inf Theory 65:5243–5259
- Guo Q, Mårtensson E, Stankovski W (2021) On the sample complexity of solving LWE using BKW-style algorithms. In: IEEE international symposium on information theory, pp 12–20
- Herold G, Kirshanova E, May A (2018) On the asymptotic complexity of solving LWE. Des Codes Cryptogr 86:55–83
- Kirchner P (2011) Improved generalized birthday attack. iacr cryptol-ogy eprint archive
- Lattice-estimator. https://github.com/malb/lattice-estimator
- Levieil E, Fouque P (2006) An improved LPN algorithm. In: Security and cryptography for networks, pp 348–359
- Lindner R, Peikert C (2011) Better key sizes (and attacks) for LWE-based encryption. In: Topics in cryptology—CTRSA'11, pp 319–339
- Liu M, Nguyen P (2013) Solving BDD by enumeration: an update. In: CT-RSA 2013, pp 293–309

- Mårtensson E (2020) Some notes on post-quantum cryptanalysis. Ph.D. thesis, Lund University
- Micciancio D, Regev O (2009) Lattice-based cryptography. In: Post-quantum cryptography, pp 147–191
- Neyman J, Pearson E (1933) On the problem of the most efficient tests of statistical hypotheses. Contain Pap Math Phys Charact 231:289–337
- Regev O (2005) On lattices, learning with errors, random linear codes, and cryptography. In: ACM symposium on theory of computing, pp 84–93
- Sage: Open source mathematics software. http://www.sagemath.org/doc/ tutorial/. Accessed 16 Mar 2023
- Schnorr C, Euchner M (1994) Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math Program 63:1–3
- TU Darmstadt Learning with Errors Challenge. https://www.latticechallenge. org/lwe_challenge/challenge. Accessed 20 July 2022
- Wagner D (2002) A generalized birthday problem. In: Advances in cryptology—CRYPTO, pp 288–304
- Zhang B, Jiao L, Wang M (2016) Faster algorithms for solving LPN. In: Advances in cryptology—EUROCRYPT, pp 168–195

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- ▶ Rigorous peer review
- Open access: articles freely available online
- ► High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at > springeropen.com