On the Complexity of the BKW Algorithm on LWE

Martin Albrecht, Carlos Cid, Jean-Charles Faugère, Fitzpatrick Robert, Ludovic Perret

On the Complexity of the BKW Algorithm on LWE

Martin R. Albrecht[1], Carlos Cid[3], Jean-Charles Faugère[2], Robert Fitzpatrick[3], and Ludovic Perret[2]

[1] Technical University of Denmark, Denmark
[2] INRIA, Paris-Rocquencourt Center, POLSYS Project
UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
CNRS, UMR 7606, LIP6, F-75005, Paris, France
[3] Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, United Kingdom
m.albrecht@mat.dtu.dk, carlos.cid@rhul.ac.uk, jean-charles.faugere@inria.fr,
robert.fitzpatrick.2010@live.rhul.ac.uk, ludovic.perret@lip6.fr

Abstract. This work presents a study of the complexity of the Blum-Kalai-Wasserman (BKW) algorithm when applied to the Learning with Errors (LWE) problem, by providing refined estimates for the data and computational effort requirements for solving *concrete* instances of the LWE problem. We apply this refined analysis to suggested parameters for various LWE-based cryptographic schemes from the literature and, as a result, provide new upper bounds for the concrete hardness of these LWE-based schemes.

1 Introduction

LWE (Learning with Errors) is a generalisation for large primes of the well-known LPN (Learning Parity with Noise) problem. It was introduced by Regev in [37] and has provided cryptographers with a remarkably flexible tool for building cryptosystems. For example, Gentry, Peikert and Vaikuntanathan presented in [21] LWE-based constructions of trapdoor functions, digital signature schemes, and identity-based encryption. Moreover, in his recent seminal work Gentry [19] resolved one of the longest standing open problems in cryptography with a construction related to LWE: the first fully homomorphic encryption scheme. This was followed by further constructions of homomorphic encryption schemes based on the LWE problem, e.g. [3, 14]. Reasons for the popularity of LWE as cryptographic primitive include its simplicity as well as convincing theoretical arguments regarding its hardness, namely, a (quantum) reduction from worst-case lattice problems, such as the Shortest Vector Problem (SVP) and Bounded Distance Decoding (BDD), to average-case LWE.

We reproduce the definition of the LWE problem from [37] where we restrict our attention to prime moduli.

Definition 1 (LWE). *Let $n \geq 1$ be a positive integer, q be an odd prime, χ be a probability distribution on $\mathbb{Z}_q$ and $\mathbf{s}$ be a secret vector in $\mathbb{Z}_q^n$. We denote by $L_{\mathbf{s},\chi}^{(n)}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ at random, choosing $e \in \mathbb{Z}_q$ according to χ, and returning $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.* LWE *is the problem of finding $\mathbf{s} \in \mathbb{Z}_q^n$ given pairs $\mathbb{Z}_q^n \times \mathbb{Z}_q$ sampled according to $L_{\mathbf{s},\chi}^{(n)}$.*

The modulus q is typically taken to be polynomial in n, and χ is the discrete Gaussian distribution on $\mathbb{Z}_q$ with mean 0 and standard deviation $\sigma = \alpha \cdot q$, for some α. Regev proved [37] that if $\sigma \geq \sqrt{n}$, then (worst-case) GAPSVP $- \tilde{\mathcal{O}}(n/\alpha)$ reduces to (average-case) LWE. This reduction is quantum when $q \in \text{poly}(n)$; it can however be made classical [35] if the modulus is super-polynomial, i.e., $q \in 2^{\tilde{\mathcal{O}}(n)}$. In this work we always assume $q \in \text{poly}(n)$.

MOTIVATION. While there is a reduction of LWE to (assumed) hard lattice problems [37], little is known about the *concrete* hardness of particular LWE instances. That is, given particular values for the prime q and the security parameter n, what is the number of bit operations required to recover the secret using currently known algorithms? As a consequence of this gap, most proposals based on LWE do not provide concrete choices for parameters and restrict themselves to asymptotic statements about security, which can be considered unsatisfactorily vague for practical purposes. In fact we see this lack of precision as one of the several obstacles to the consideration of LWE-based schemes for real-world applications.

RELATED WORK. Since LWE can be reduced to hard lattice problems, advances in and concrete estimates for lattice algorithms typically carry over to LWE. Indeed, the expected complexity of lattice algorithms is often exclusively considered when parameters for LWE-based schemes are discussed. However, while the effort on improving lattices algorithms is intense [39, 15, 33, 18, 34, 24, 32, 36], direct algorithms for tackling the LWE problem remain rarely investigated from an algorithmic point of view. For example, the main subject of this paper – the BKW algorithm – specifically applied to the LWE problem has so far received no treatment in the literature[1]. However, we note that the BKW algorithm can be viewed as an oracle producing short vectors in the dual lattice spanned by the $\mathbf{a}_i$ and hence shares some similarities with combinatorial SVP solvers. Furthermore, it is only recently that a non lattice-based alternative to BKW has been proposed for LWE: Arora and Ge [5] proposed a new algebraic technique for solving LWE, with total complexity (time and space) of $2^{\tilde{\mathcal{O}}(\sigma^2)}$ (it is thus subexponential when $\sigma < \sqrt{n}$, remaining exponential when $\sigma \geq \sqrt{n}$). It is worth noting that Arora and Ge achieve the $\sqrt{n}$ hardness-threshold found by Regev [37], but with a constructive approach. We note however that currently the main relevance of Arora-Ge's algorithm is asymptotic; it is an open question whether one can improve its practical efficiency.

For comparison, the situation is much different in code-based cryptography. That is, improvements on the Information Set Decoding (ISD) algorithm – the classical technique for decoding random linear codes – are continuously reported, e.g. [16, 9, 10, 30, 8], allowing to rather easily determine concrete parameters for code-based schemes. From a more general perspective, we emphasise that improving the constants of exponential algorithms solving hard computational problems is emerging as a new important research area in computer science. For computational problems related to cryptography, we mention recent results on solving knapsacks [7, 25], solving sets of non-linear equations [11, 12], as well as lattice problems.

CONTRIBUTION. We present a detailed study of a dedicated version of the Blum, Kalai and Wasserman (BKW) algorithm [13] for LWE with discrete Gaussian noise. To our knowledge, this is the first time that such a detailed description appears in the literature. Given an instance of the LWE problem as described in Definiton 1, the BKW algorithm can be viewed as consisting of three stages: sample reduction, hypothesis testing to recover a subset of the secret, and back substitution. By studying in detail each of these stages, we take the first steps to 'de-asymptotic-ify' our understanding of the hardness of LWE under the BKW algorithm. That is, by investigating the exact complexity of the algorithm, we provide concrete values for the expected number of bit operations for solving instances of the LWE problem. The BKW algorithm is known to have complexity $2^{\mathcal{O}(n)}$ when applied to LWE instances with a prime modulus polynomial in n [37]; in this paper we provide both the leading constant of the exponent in $2^{\mathcal{O}(n)}$ and concrete costs of BKW when applied to LWE. More precisely, we first show the following theorem in Section 3.

Theorem 1 (informal). *Let $(\mathbf{a}_i, c_i)$ be samples following $L^{(n)}_{\mathbf{s},\chi}$, set $a = \lfloor \log_2(1/(2\alpha)^2) \rceil$, $b = n/a$. Let d be a small constant $0 < d < \log_2(n)$. Assume α is such that $q^b = q^{n/a} = q^{n/\log_2(1/(2\alpha)^2)}$ is superpolynomial in n. Then, given these parameters, the cost of the BKW algorithm to recover $\mathbf{s}$ is*

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

[1] However, a detailed study of the algorithm to the LPN-case was provided [17], which in fact heavily inspired this work.

operations in $\mathbb{Z}_q$*. Furthermore,*

$$a \cdot (\lceil q^b/2 \rceil) + \text{poly}(n)$$

calls to $L^{(n)}_{\mathbf{s},\chi}$ *are needed.*

We discuss variants of the algorithm in Section 4, while in Section 5 we focus on applying BKW to various parameter choices for LWE from the literature [37, 31, 28, 3]. Furthermore, we report experimental results for small instances of LWE (Section 6) and discuss alternative approaches (Section 7).

2 Preliminaries

Let $\mathcal{N}(\mu, \sigma^2)$ denote the Gaussian distribution with mean μ and standard deviation σ. The LWE problem considers the discrete Gaussian distribution over $\mathbb{Z}_q$ with mean 0. This distribution can be obtained by *discretising* the corresponding *wrapped* Gaussian distribution. To wrap $\mathcal{N}(0, \sigma^2)$ mod q, we denote by $p(\phi)$ the probability density function determined by σ, define the periodic variable $\theta := \phi \bmod q$ and let

$$p'(\theta) = \sum_{k=-\infty}^{\infty} p(\theta + qk) \text{ for } -q/2 < \theta \leq q/2. \tag{1}$$

As $|k|$ increases, the contribution of $p(\theta + qk)$ falls rapidly; in fact, exponentially fast. Hence, we can pick a point at which we 'cut' $p'(\theta)$ and work with this approximation. We denote the distribution sampled according to p' and rounded to the nearest integer in the interval $]\frac{-q}{2}, \frac{q}{2}]$ by $\chi_{\alpha,q}$, where $\sigma = \alpha \cdot q$. That is,

$$\Pr[X = x] = \int_{x-\frac{1}{2}}^{x+\frac{1}{2}} p'(t)dt \tag{2}$$

We note that we can explicitly compute $\Pr[X = x]$ because both q and k are poly(n).

We remark that in [17], the authors assume that the samples used for hypothesis-testing are independent – this is not always the case due to the combined noise elements for reduced samples often sharing certain of the original noise elements. In practice, this does not have a noticeable impact and the covariance of such samples is generally very small. Thus we ignore such effects and assume firstly that our final samples have zero covariance (in general the covariance matrix for the noise of the final samples will be sparse) and that we can treat the samples as being independent. We note that, during the course of extensive experimentation, behaviour deviating from that implied by these assumptions was never observed.

As partial justification for these assumptions, for a heuristic estimate of the sparsity of the covariance matrix of our final samples, we observe that we have the following: after the first stage, consider any two samples. Each sample has a set of 2^a added error elements implicit, with each error element having been 'chosen' at random from (at most) $(q^b - 1)/2$ possible error elements. If we take two such sets, then the probability that their intersection is non-empty (and hence their covariance is non-zero) is (assuming we discount the possibility of collisions within each set) roughly equal to $(2^{a+1})/(q^b - 1)$, which will be small for all reasonable choices of parameters.

We state a straightforward lemma which will be useful in our computations later.

Lemma 1. *Let* $X_1, X_1, \ldots, X_m$ *be independent random variables, with* $X_i \sim \mathcal{N}(\mu, \sigma^2)$*. Then their sum* $X = \sum_{i=1}^{m} X_i$ *is also normally distributed, with* $X \sim \mathcal{N}(m\mu, m\sigma^2)$*.*

One can show that a similar result also holds when $X_i \sim \chi_{\alpha,q}$.

We also explain our notation: we always start counting at zero, and denote vectors in bold. Given a vector $\mathbf{a}$, we denote by $\mathbf{a}_{(i)}$ the i-th entry in $\mathbf{a}$, i.e., a scalar, and by $\mathbf{a}_{(i,j)}$ the subvector of $\mathbf{a}$ spanning the entries at indices $i, \ldots, j-1$. When given a list of vectors, we index its elements by subscript, e.g., $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2$, to denote the first three vectors of the list. When we write $(\mathbf{a}_i, c_i)$ we always mean the output of an oracle which should be clear from the context. In particular, $(\mathbf{a}_i, c_i)$ does not necessarily refer to samples following the distribution $L^{(n)}_{\mathbf{s},\chi}$.

3 The BKW Algorithm

The BKW algorithm was proposed by Blum, Kalai and Wasserman [13] as a method for solving the LPN problem, with sub-exponential complexity, requiring $2^{\mathcal{O}(n/\log n)}$ samples and time. The algorithm can be adapted for tackling the LWE problem, with complexity $2^{\mathcal{O}(n)}$, when the modulus is taken to be polynomial in n. The BKW algorithm can be viewed as consisting of three stages: (a) sample reduction, (b) hypothesis testing to recover a subset of the secret and (c) back substitution. On a high level, the first stage of the BKW algorithm can be described as a form of Gaussian elimination which, instead of treating each column independently, considers 'blocks' of b columns per iteration, where b is a parameter of the algorithm. Following this reduction, the second stage performs hypothesis tests to recover components of the secret vector $\mathbf{s}$. Finally, the third stage performs back substitution such that the whole process can be continued on a smaller instance of LWE. The main idea of the algorithm is to minimise the number of row operations (additions) in the first stage, as this has a strong influence in the number of samples required in the later stages for reliably recovering each of the components of $\mathbf{s}$.

The way we study the complexity of the BKW algorithm for solving the LWE problem is closely related to the method described in [17]: given an oracle that returns samples according to the probability distribution $L^{(n)}_{\mathbf{s},\chi}$, we use the algorithm's first stage to construct an oracle returning samples according to another distribution, which we call $B^{(n)}_{\mathbf{s},\chi,a}$, where $a = \lceil n/b \rceil$ denotes the number of 'levels' of addition. The complexity of the algorithm is related to the number of operations performed in this transformation, to obtain the required number of samples for hypothesis testing. We now study the complexity of the first stage of the BKW algorithm.

3.1 Elimination

Given $n \in \mathbb{Z}$, select a positive integer $b \leq n$ (the window width), and let $a := \lceil n/b \rceil$ (the addition depth). Given an LWE oracle (which by abuse of notation, we will also denote by $L^{(n)}_{\mathbf{s},\chi}$), we denote by $B^{(n)}_{\mathbf{s},\chi,\ell}$ a related oracle which outputs samples where the first $b \cdot \ell$ columns of each $\mathbf{a}_i$ are zero, generated under the distribution (which again by abuse of notation, we denote by $B^{(n)}_{\mathbf{s},\chi,\ell}$) obtained as follows:

- if $\ell = 0$, then $B^{(n)}_{\mathbf{s},\chi,0}$ is simply $L^{(n)}_{\mathbf{s},\chi}$;
- if $0 < \ell < a$, the distribution $B^{(n)}_{\mathbf{s},\chi,\ell}$ is obtained by taking the difference of two vectors from $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ that agree on the elements $(\mathbf{a}_{((\ell-1)\cdot b)}, \mathbf{a}_{((\ell-1)\cdot b+1)}, \ldots, \mathbf{a}_{(\ell\cdot b-1)})$.

We can then describe the first stage of the BKW algorithm as the (recursively constructed) series of sample oracles $B^{(n)}_{\mathbf{s},\chi,\ell}$, for $0 \leq \ell < a$. Indeed, we define $B^{(n)}_{\mathbf{s},\chi,0}$ as the oracle which simply returns samples from $L^{(n)}_{\mathbf{s},\chi}$, while $B^{(n)}_{\mathbf{s},\chi,\ell}$ is constructed from $B^{(n)}_{\mathbf{s},\chi,\ell-1}$, for $\ell \geq 1$. We will make use of a set of tables T (maintained across

oracle calls) to store (randomly-chosen) vectors that will be used to reduce samples arising from our oracles. More explicitly, given a parameter $b \leq n$ for the window width, and letting $a = \lceil n/b \rceil$, we can describe the oracle $B^{(n)}_{\mathbf{s},\chi,\ell}$ as follows:

1. For $\ell = 0$, we can obtain samples from $B^{(n)}_{\mathbf{s},\chi,0}$ by simply calling the LWE oracle $L^{(n)}_{\mathbf{s},\chi}$ and returning the output.
2. For $\ell = 1$, we repeatedly query the oracle $B^{(n)}_{\mathbf{s},\chi,0}$ to obtain (up to) $(q^b - 1)/2$ samples $(\mathbf{a}, c)$ with distinct non-zero vectors for the first b columns of $\mathbf{a}$. We only collect $(q^b - 1)/2$ such vectors because we exploit the symmetry of the finite field representation and that of the noise distribution. We use these samples to populate the table T^1, indexed by the first b entries of $\mathbf{a}$. During this process, whenever we obtain a sample $(\mathbf{a}', c')$ from $B^{(n)}_{\mathbf{s},\chi,0}$ with its first b entries either all equal to zero, or that are already in T^1 (say, $(\mathbf{a}, c)$), we return either $(\mathbf{a}', c')$ or $(\mathbf{a}' \pm \mathbf{a}, c' \pm c)$, respectively, as a sample from $B^{(n)}_{\mathbf{s},\chi,1}$. Further calls to the oracle $B^{(n)}_{\mathbf{s},\chi,1}$ proceeds in similar manner, but using (and potentially adding entries to) the same table T^1.
3. for $1 < \ell < a$, we proceed as above: we make use of the table T^ℓ (constructed by calling $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ up to $(q^b - 1)/2$ times) to reduce any output sample from $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ which has the b entries in its ℓ-th block already in T^ℓ, to generate a sample from $B^{(n)}_{\mathbf{s},\chi,\ell}$.

Pseudo-code for the oracle $B^{(n)}_{\mathbf{s},\chi,\ell}$, for $0 < \ell < a$, is given in Algorithm 1.

Input: b – an integer $0 < b \leq n$
Input: ℓ – an integer $0 < \ell < a$
begin
  $T^\ell \leftarrow$ array indexed by $\mathbb{Z}_q^b$ maintained across all runs of $B^{(n)}_{\mathbf{s},\chi,\ell}$;
  query $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ to obtain $(\mathbf{a}, c)$;
  if $\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)} =$ *all zero vector* **then**
    return $(\mathbf{a}, c)$;
  while $T_{\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}} = \varnothing$ **do**
    $T_{\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}} \leftarrow (\mathbf{a}, c)$;
    $T_{-\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}} \leftarrow (-\mathbf{a}, -c)$;
    query $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ to obtain $(\mathbf{a}, c)$;
    if $\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)} =$ *all zero vector* **then**
      return $(\mathbf{a}, c)$;
  $(\mathbf{a}', c') \leftarrow T_{\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}}$;
  return $(\mathbf{a} - \mathbf{a}', c - c')$;

Algorithm 1: $B^{(n)}_{\mathbf{s},\chi,\ell}$ for $0 < \ell < a$

Then, given an LWE oracle $L^{(n)}_{\mathbf{s},\chi}$ outputting samples of the form $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, where $\mathbf{a}, \mathbf{s} \in \mathbb{Z}_q^n$, the oracle $B^{(n)}_{\mathbf{s},\chi,a-1}$ can be seen as another LWE oracle outputting samples of the form $(\mathbf{a}', \langle \mathbf{a}', \mathbf{s}' \rangle + e')$, where $\mathbf{a}', \mathbf{s}' \in \mathbb{Z}_q^k$, with $k = n \mod b$, if b does not divide n, or $k = b$ otherwise, and e' is generated with a different distribution (related to the original error distribution and the value a). The vector $\mathbf{s}'$ is defined to be the last k components of $\mathbf{s}$. For the remainder of this section we will assume that $n \mod b = 0$, and therefore $k = b$ (this is done to simplify the notation, but all the results obtained can be easily adapted when the last block has length $k < b$).

Moreover, in this work we introduce an additional parameter $d < b$ which does not exist in the original BKW [13]. This parameter is used to reduce the number of hypotheses that need to be tested in the second stage of the algorithm, and arises from the fact we work with primes $q > 2$. At times, instead of working with the last block of length b, which could lead to potentially exponentially many hypotheses q^b to be tested, we may employ a final reduction phase – $B^{(n)}_{\mathbf{s},\chi,a}$ – to reduce the samples to $d < b$ non-zero entries in $\mathbf{a}$. Thus d will represent the number of columns in our final block, i.e., the number of elements of the secret over which we conduct hypothesis tests. So after running the $B^{(n)}_{\mathbf{s},\chi,a-1}$ algorithm, we may decide to split the final block to obtain vectors over $\mathbb{Z}_q^d$. If so, we run the reduction function described above once more, which we will denote by $B^{(n)}_{\mathbf{s},\chi,a}$. In the simple case where we do not split the last block (i.e., $d = b$), we adopt the convention that we will also call the $B^{(n)}_{\mathbf{s},\chi,a}$ function, but it will perform no extra action (i.e., it simply calls $B^{(n)}_{\mathbf{s},\chi,a-1}$). Thus we have that for a choice of $1 \leq d \leq b$, the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ will output samples of the form $(\mathbf{a}', \langle \mathbf{a}', \mathbf{s}' \rangle + e')$, where $\mathbf{a}', \mathbf{s}' \in \mathbb{Z}_q^d$.

From the constructions above, it follows that the cost of one call to $B^{(n)}_{\mathbf{s},\chi,\ell}$ is at most $\lceil q^b/2 \rceil$ calls to $B^{(n)}_{\mathbf{s},\chi,\ell-1}$. We also need at most one addition of two outputs of $B^{(n)}_{\mathbf{s},\chi,\ell-1}$, which have the first $b \cdot \ell$ entries in common. This operation requires $n + 1 - b \cdot \ell$ additions in $\mathbb{Z}_q$. Furthermore, since we maintain T^ℓ across different runs of $B^{(n)}_{\mathbf{s},\chi,\ell}$, this cost is amortised across different runs of $B^{(n)}_{\mathbf{s},\chi,\ell}$. Hence, when $\ell > 0$ the cost of calling $B^{(n)}_{\mathbf{s},\chi,\ell}$ a total of m times is upper bounded by:

$$m + \frac{q^b - 1}{2} \text{ calls to } B^{(n)}_{\mathbf{s},\chi,\ell-1} \text{ and } m \text{ additions of outputs of } B^{(n)}_{\mathbf{s},\chi,\ell-1}.$$

Overall we obtain Lemma 2.

Lemma 2. *Let $n \geq 1$ be the dimension of the LWE secret vector, q be a modulus, and $d, b \in \mathbb{Z}$ with $1 \leq d \leq b \leq n$, and define $a = \lceil n/b \rceil$. The cost of calling $B^{(n)}_{\mathbf{s},\chi,a}$, which returns samples $(\mathbf{a}_i, c_i)$ with at most the d rightmost entries of $\mathbf{a}_i$ non-zero, m times is upper bounded by*

$$\left(\frac{q^b-1}{2}\right) \cdot \left(\frac{a(a-1)}{2} \cdot (n+1) - \frac{ba(a-1)}{4} - \frac{b}{6}\left((a-1)^3 + \frac{3}{2}(a-1)^2 + \frac{1}{2}(a-1)\right)\right) + m \cdot \left(\frac{a}{2} \cdot (n+2)\right)$$
$$< \left(\frac{q^b-1}{2}\right) \cdot \left(\frac{a(a-1)}{2} \cdot (n+1)\right) + m \cdot (\frac{a}{2} \cdot (n+2))$$

additions in $\mathbb{Z}_q$ and $a \cdot (\lceil q^b/2 \rceil) + m$ calls to $L^{(n)}_{\mathbf{s},\chi}$.

Proof. We may think of the $B^{(n)}_{\mathbf{s},\chi,\ell}$ oracles as constructing the matrix

$$B = \begin{pmatrix}
T^1 & & & & & & & & & \\
0 \ \ldots 0 & T^2 & & & & & & & & \\
0 \ \ldots 0 & 0 \ \ldots 0 & T^3 & & & & & & & \\
\vdots \ \ddots \ \vdots & \vdots \ \ddots \ \vdots & \vdots & \ddots & & & & & & \\
0 \ \ldots 0 & 0 \ \ldots 0 & 0 \ \ldots 0 & 0 \ \ldots 0 & T^{a-1} & & & & & \\
0 \ \ldots 0 & 0 \ \ldots 0 & 0 \ \ldots 0 & 0 \ \ldots 0 & 0 & T^a & & & & \\
0 \ \ldots 0 & 0 \ \ldots 0 & 0 \ \ldots 0 & 0 \ \ldots 0 & \ldots & 0 & M & & &
\end{pmatrix}$$

where, for $1 \leq i \leq a-1$, T^i represents a submatrix of $(q^b - 1)/2$ rows and $n + 1 - b \cdot (i-1)$ columns (the $+1$ accounts for the c_i column). According to our convention, T^a is either a submatrix of $(q^{b-d} - 1)/2$ rows and $n + 1 - b \cdot (a-1)$ columns if we split the last block (i.e. $d < b$), or an empty matrix. Finally M represents a submatrix with m rows and $d + 1$ columns. The matrix B has therefore at most $(a-1) \cdot ((q^b - 1)/2) +$

$((q^{b-d}-1)/2)+m < a\cdot(\lceil q^b/2\rceil)+m$ rows and hence needs as many calls to $L^{(n)}_{\mathbf{s},\chi}$ to be constructed. This proves the second claim.

For the upper-bound on the number of additions necessary in $\mathbb{Z}_q$, we have the following (we treat the worst case, where full construction of all T tables is necessary before obtaining any samples from $B^{(n)}_{\mathbf{s},\chi,a}$): The construction of a T-tables is required, only $a-1$ (at most) of which require additions.

1. The construction of table T^1 requires 0 ring additions.
2. The construction of table T^2 requires at most $((q^b-1)/2)\cdot(n+1-b)$ additions.
3. The construction of table T^3 requires at most $((q^b-1)/2)\cdot((n+1-b)+(n+1-2b))$ additions.
4. In general, for $2<i<a$, the construction of table T^i requires at most

$$\left(\frac{q^b-1}{2}\right)\cdot\left((i-1)\cdot(n+1)-\sum_{j=1}^{i-1}j\cdot b\right)=\left(\frac{q^b-1}{2}\right)\cdot\left((i-1)\cdot(n+1)-\frac{i(i-1)}{2}\cdot b\right)$$
$$=\left(\frac{q^b-1}{2}\right)\cdot(i-1)\cdot\left((n+1)-\frac{i}{2}\cdot b\right).$$

5. The construction of T^a - the above expression is an upper bound for $i=a$.
6. Thus, the construction of all the T^i tables requires at most

$$\left(\frac{q^b-1}{2}\right)\cdot\sum_{j=2}^{a}\left((j-1)\cdot((n+1)-\frac{j}{2}\cdot b)\right)=\left(\frac{q^b-1}{2}\right)\cdot\sum_{k=1}^{a-1}\left(k\cdot(n+1)-\frac{k(k+1)}{2}\cdot b\right)$$
$$=\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)-\sum_{k=1}^{a-1}\frac{k(k+1)}{2}\cdot b\right)$$
$$=\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)-\frac{b}{2}\left(\sum_{k=1}^{a-1}k^2+k\right)\right)$$
$$=\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)-\frac{ba(a-1)}{4}-\frac{b}{2}\left(\sum_{k=1}^{a-1}k^2\right)\right)$$
$$=\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)-\frac{ba(a-1)}{4}-\frac{b}{6}\left((a-1)^3+\frac{3}{2}(a-1)^2+\frac{1}{2}(a-1)\right)\right)$$

   additions in $\mathbb{Z}_q$.
7. Now, for the construction of our m final samples, the construction of each of these samples requires at most

$$(n+1-b)+(n+1-2b)+\ldots+(n+1-a\cdot b)=\sum_{i=1}^{a}(n+1-ib)$$
$$=a\cdot(n+1)-\frac{a(a+1)}{2}\cdot b$$
$$<a\cdot\left((n+1)-\frac{n}{2}\right)$$
$$=\frac{a}{2}\cdot(n+2)$$

   additions (in $\mathbb{Z}_q$).

8. Thus, the number of additions (in $\mathbb{Z}_q$) incurred through calling $B^{(n)}_{\mathbf{s},\chi,a}$ m times is upper-bounded by:

$$\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)-\frac{ba(a-1)}{4}-\frac{b}{6}\left((a-1)^3+\frac{3}{2}(a-1)^2+\frac{1}{2}(a-1)\right)\right)+m\cdot\left(\frac{a}{2}\cdot(n+2)\right)$$

and this concludes the proof of the lemma. □

The memory requirements for storing the tables T^i are established in Lemma 3 below.

Lemma 3. *Let $n \geq 1$ be the number of variables, q be a modulus, and $d, b \in \mathbb{Z}$ with $1 \leq d \leq b \leq n$, and define $a = \lceil n/b \rceil$. The memory required to store the table T^i is upper bounded by*

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

finite field elements.

Proof. The table T^1 has $\frac{q^b}{2}$ entries each of which holds $n+1$ finit field elements. The table T^2 has the same number of entries but holds on $n+1-b$ finite field elements. Overall, we get that all tables together hold

$$\begin{aligned}
\sum_{i=1}^{a}\left(\frac{q^b}{2}\right)\cdot\left(n+1-(i-1)b\right) &= \left(\frac{q^b}{2}\right)\sum_{i=1}^{a} n+1-(i-1)b \\
&= \left(\frac{q^b}{2}\right)\cdot\left(a(n+1+b)-\sum_{i=1}^{a} ib\right) \\
&= \left(\frac{q^b}{2}\right)\cdot\left(a(n+1+b)-b\frac{(a+1)a}{2}\right) \\
&= \left(\frac{q^b}{2}\right)\cdot a\cdot\left(n+1+b-b\frac{a+1}{2}\right) \\
&= \left(\frac{q^b}{2}\right)\cdot a\cdot\left(n+1-b\frac{a-1}{2}\right)
\end{aligned}$$

finite field elements. □

Note however that, unlike the original LWE oracle $L^{(n)}_{\mathbf{s},\chi}$, the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ may output zero vectors, which will offer no information for the hypothesis tests. In particular, calling it m times does not guarantee that we get m (useful) samples with non-zero coefficients in $\mathbf{a}_i$. We assume the probability of a zero vector being output by $B^{(n)}_{\mathbf{s},\chi,a}$ is $\frac{1}{q^d-1}$, and thus expect to have to call the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ around $\frac{q^d}{q^d-1}\cdot m$ times to obtain $\approx m$ useful samples with good probability.

We now consider the noise distribution associated with samples obtained from the oracles $B^{(n)}_{\mathbf{s},\chi,i}$, $0 \leq i \leq a$. If we have that the noise distribution associated with samples from $L^{(n)}_{\mathbf{s},\chi}$ is $\chi = \chi_{\alpha,q}$, then it follows from Lemmas 1 and 2 that the noise distribution of samples obtained from $B^{(n)}_{\mathbf{s},\chi,\ell}$ follows $\chi_{\sqrt{2^\ell}\alpha,q}$. For the sake of simplicity, we denote this distribution by χ_ℓ in the remainder of this work. Also for the sake of simplicity, we assume that the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ performs non-trivial operations on the output of $B^{(n)}_{\mathbf{s},\chi,a-1}$, i.e., the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ performs a further reduction step. In other words we assume that the final oracle $B^{(n)}_{\mathbf{s},\chi,a}$ results in a further increase in the standard deviation of the noise distribution associated with the final samples which are used to test hypotheses for elements of $\mathbf{s}$.

If we let $\mathbf{s}' := \mathbf{s}_{(n-d,n)} = (\mathbf{s}_{(n-d)}, \ldots, \mathbf{s}_{(n-1)})$, then the output of $B^{(n)}_{\mathbf{s},\chi,a}$ is generated as

$$\mathbf{a} \leftarrow_{\$} \mathbb{Z}_q^d, e \leftarrow_{\$} \chi_a : (\mathbf{a}, \langle \mathbf{a}, \mathbf{s}' \rangle + e).$$

3.2 Hypothesis-Testing

To give concrete estimates for the time and data complexity of solving an LWE instance using BKW, we formulate the problem of solving an LWE instance as the problem of distinguishing between two different distributions. Assume we have m samples from $B^{(n)}_{\mathbf{s},\chi,a}$, then focusing on the last d columns, we clearly have q^d possible hypotheses to test, each one corresponding to an assignment of values to the last d elements of $\mathbf{s}$. In what follows, we examine each hypothesis in turn and derive a hypothesised set of noise values as a result (each one corresponding to one of the m samples). We show that if we have guessed incorrectly for the subvector $\mathbf{s}'$ of $\mathbf{s}$ then the distribution of these hypothesised noise elements will be (almost) uniform while if we guess correctly then these hypothesised noise elements will be distributed according to χ_a.

Recall that the last window we consider contains elements $\in \mathbb{Z}_q^d$, hence $B^{(n)}_{\mathbf{s},\chi,a}$ returns samples of the form $(\mathbf{a}_i, c_i)$ with $\mathbf{a}_i \in \mathbb{Z}_q^d$ and $c_i \in \mathbb{Z}_q$. For our hypothesis-testing strategies, we think of each of these samples as giving rise to many equations

$$f_i = -c_i \pm j + \sum_{k=0}^{d-1} (\mathbf{a}_i)_{(k)} x_{(k)} \text{ for } 0 \leq j < q/2.$$

Given a number of these samples, in order to get an estimate for $\mathbf{s}'$, we run through q^d hypotheses and compute an array of scores S indexed by the possible guesses in $\mathbb{Z}_q^d$. That is, a function W assigns a weight to elements in $\mathbb{Z}_q$ which represent the noise under the hypothesis $\mathbf{s}' = \mathbf{v}$. For each guess $\mathbf{v}$ we sum over the weighted noises $W(-c_i + \sum_{k=0}^{d-1} (\mathbf{a}_i)_{(k)} \cdot v_{(k)})$. If W is such that the counter $S_{\mathbf{v}}$ grows proportionally to the likelihood that $\mathbf{v}$ is the correct guess, then the counter $S_{\mathbf{s}'}$ will grow fastest. Pseudo-code is given in Algorithm 2.

Input: F – a set of m samples following $B^{(n)}_{\mathbf{s},\chi,a}$
Input: W – a weight function mapping members of $\mathbb{Z}_q$ to real numbers
begin
  $S \leftarrow$ array filled with zeros indexed by $\mathbb{Z}_q^d$;
  for $\mathbf{v} \in \mathbb{Z}_q^d$ **do**
    $w_{\mathbf{v}} \leftarrow \emptyset$;
    for $f_i \in F$ **do**
      write f_i as $-c_i + \sum_{k=0}^{d-1} (\mathbf{a}_i)_{(k)} \cdot x_{(k)}$;
      $j \leftarrow \langle \mathbf{a}_i, \mathbf{v} \rangle - c_i$;
      $w_{\mathbf{v}} \leftarrow w_{\mathbf{v}} \cup \{W(j)\}$;
    $S_{\mathbf{v}} \leftarrow \sum_{w_i \in w_{\mathbf{v}}} w_i / m$;
  return S

Algorithm 2: Analysing candidates.

Lemma 4. *Running hypothesis testing costs $m \cdot q^d$ operations in $\mathbb{Z}_q$.*

Proof (Sketch). Evaluating $\langle \mathbf{a}_i, \mathbf{v} \rangle - c_i$ naively costs $2d$ operations in $\mathbb{Z}_q$. However, reorganising the loop and iterating over $\mathbb{Z}_q^d$ using a Gray code [22] generalised to $\mathbb{Z}_q$ allows to reduce this to 1 operation in $\mathbb{Z}_q$. Hence, Algorithm 2 costs $m \cdot q^d$ operations in $\mathbb{Z}_q$. □

Recall that χ_a is the distribution of the errors under a right guess. Now, let $\mathcal{U}_a$ denote the distribution of errors under a wrong guess $\mathbf{v} \neq \mathbf{s}'$. By the Neyman-Pearson Leamma, the most powerful test of whether samples follow one of two known distributions is the log-likelihood ratio.

Hence, for j, with $\lceil -q/2 \rceil \leq j \leq \lfloor q/2 \rfloor$, we set:

$$W(j) := \log_2 \left(\frac{\Pr[e \leftarrow_\$ \chi_a : e = j]}{\Pr[e \leftarrow_\$ \mathcal{U}_a : e = j]} \right). \tag{3}$$

Remark 1. Regev [37] describes a distinguisher based on the periodicy of the two noise distributions. We experimentally verified that such distinguisher does not improve our results.

Next, we establish the relation between $\tilde{p}_j := \Pr[e \leftarrow_\$ \mathcal{U}_a : e = j]$ and $p_j := \Pr[e \leftarrow_\$ \chi_a : e = j]$.

Lemma 5. *Given a wrong guess $\mathbf{v}$ for $\mathbf{s}'$, for each element $f_i = -c_i + \sum_{k=0}^{d-1} (\mathbf{a}_i)_{(k)} x_{(k)} \in F$, with $c_i = \langle \mathbf{a}_i, \mathbf{s}' \rangle - e_i$, the probability of error j appearing is:*

$$\tilde{p}_j := \Pr[e \leftarrow_\$ \mathcal{U}_a : e = j] = \frac{q^{d-1} - p_j}{q^d - 1}. \tag{4}$$

Proof. We write $\mathbf{v} = \mathbf{s}' + \mathbf{t}$. Since $\mathbf{v}$ is a wrong guess, we must have $\mathbf{t} \neq \mathbf{0}$. For a fixed $\mathbf{s}'$ and for $\mathbf{a}_i, \mathbf{t} \neq \mathbf{0}$, it holds that:

$$\begin{aligned} \tilde{p}_j &= \Pr[e \leftarrow_\$ \mathcal{U}_a : e = j] = \Pr[\langle \mathbf{a}_i, \mathbf{v} \rangle - c_i = j] = \Pr[\langle \mathbf{a}_i, \mathbf{t} \rangle - e_i = j] \\ &= \sum_{y=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} \left(\Pr[\sum_{k=0}^{d-1} \mathbf{a}_{i(k)} \mathbf{t}_{(k)} = y] \cdot \Pr[j + e_i = y] \right). \end{aligned}$$

This is equal to:

$$\begin{aligned} & \textstyle\sum_{y=\lceil -q/2 \rceil}^{y=-1} \left(\Pr[\sum_{k=0}^{d-1} \mathbf{a}_{i(k)} \mathbf{t}_{(k)} = y] \cdot \Pr[j + e_i = y] \right) \\ + & \textstyle\sum_{y=1}^{y=\lfloor q/2 \rfloor} \left(\Pr[\sum_{k=0}^{d-1} \mathbf{a}_{i(k)} \mathbf{t}_{(k)} = y] \cdot \Pr[j + e_i = y] \right) \\ + & \Pr[\sum_{k=0}^{d-1} \mathbf{a}_{i(k)} \mathbf{t}_{(k)} = 0] \cdot \Pr[e_i = -j]. \end{aligned}$$

Now, since our q is prime, for any two non-zero elements $y, z \in \mathbb{Z}_q$, we have that:

$$\Pr[\sum_{k=0}^{d-1} \mathbf{a}_{i(k)} \mathbf{t}_{(k)} = y] = \Pr[\sum_{k=0}^{d-1} \mathbf{a}_{i(k)} \mathbf{t}_{(k)} = z].$$

We denote this probability by $p_{(\neq 0)}$. Conversely, we denote the probability of obtaining $\langle \mathbf{a}_i, \mathbf{t} \rangle = 0$ by $p_{(=0)}$. Then we clearly have $p_{(\neq 0)} = \frac{1 - p_{(=0)}}{q-1}$. Thus we can write:

$$\begin{aligned} \tilde{p}_j = & \left(p_{(\neq 0)} \sum_{y=\lceil -q/2 \rceil}^{y=-1} \Pr[j + e_i = y] \right) + \left(p_{(\neq 0)} \sum_{y=\lceil -q/2 \rceil}^{y=-1} \Pr[j + e_i = y] \right) \\ & + \Pr[\sum_{k=0}^{d-1} \mathbf{a}_{i(k)} \mathbf{t}_{(k)} = 0] \cdot \Pr[e_i = -j]. \end{aligned}$$

Remark that

$$1 - p_{-j} = \Pr[e_i \leftarrow_\$ \chi_a : e_i \neq -j] = \sum_{y=\lceil -q/2 \rceil}^{y=-1} \Pr[j + e_i = y] + \sum_{y=1}^{\lfloor q/2 \rfloor} \Pr[j + e_i = y].$$

By definition, $p_{-j} = p_j$. Thus:

$$\tilde{p}_j = p_{(\neq 0)} \cdot (1 - p_j) + p_{(=0)} \cdot p_j.$$

Now, to determine $p_{(=0)}$, the exclusion of zero-vectors from the set of all possible dot products reduces the number of zero dot products from $q^d + q^{2d-1} - q^{d-1}$ to $q^{2d-1} - q^{d-1} - q^d + 1$. Thus we have that the probability of obtaining a zero dot product is:

$$p_{(=0)} = \frac{q^{2d-1} - q^{d-1} - q^d + 1}{q^{2d} - 2q^d + 1} = \frac{q^{d-1} - 1}{q^d - 1}.$$

Thus, we have:

$$\tilde{p}_j = (1 - p_j) \cdot \frac{q^{d-1}}{q^d - 1} + p_j \cdot \frac{q^{d-1} - 1}{q^d - 1} = \frac{q^{d-1} - p_j}{q^d - 1}$$

as required. □

For the final backsubstitution stage, we wish to ensure that the score for the correct guess $\mathbf{v} = \mathbf{s}'$ is highest among the entries of S. Thus, what remains to be established is the size $m = |F|$ needed such that the score for the right guess $\mathbf{v} = \mathbf{s}'$ is the highest. Under our sample independence assumptions, by the Central Limit theorem, the distribution of $S_{\mathbf{v}}$ approaches a Normal distribution as m increases. Hence, for sufficiently large m we assume that we may approximate the discrete distribution $S_{\mathbf{v}}$ by a normal distribution [6]. If $\mathcal{N}(\mu, \sigma^2)$ denotes a Normal distribution with mean μ and standard deviation σ we denote the distribution for $\mathbf{v} = \mathbf{s}'$ by $D_c = \mathcal{N}(\mathrm{E}_c, \mathrm{Var}_c)$ and for $\mathbf{v} \neq \mathbf{s}'$ by $D_w = \mathcal{N}(\mathrm{E}_w, \mathrm{Var}_w)$.

Establishing m hence first of all means establishing $\mathrm{E}_c, \mathrm{E}_w, \mathrm{Var}_c$ and Var_w. We start with E_c.

Lemma 6. *Let* $(\mathbf{a}_0, c_0), \ldots, (\mathbf{a}_{m-1}, c_{m-1})$ *be samples following* $B^{(n)}_{\mathbf{s},\chi,a}$, $\mathbf{v} \in \mathbb{Z}_q^d$, $p_j := \Pr(e \leftarrow_\$ \chi_a : e = j)$, $w_j := W(j)$ *and* $S_{\mathbf{v}} = \frac{1}{m} \sum_{i=0}^{m-1} W(\langle \mathbf{a}_i, \mathbf{v} \rangle - c_i)$. *When* $\mathbf{v} = \mathbf{s}'$, $\mathrm{E}(S_{\mathbf{v}})$ *is given by:*

$$\mathrm{E}_c = \mathrm{E}(S_{\mathbf{v}} \mid \mathbf{v} = \mathbf{s}') = \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} p_j w_j = p_0 w_0 + 2 \cdot \sum_{j=1}^{j=\lfloor q/2 \rfloor} p_j w_j. \tag{5}$$

Proof. First, we remark that:

$$\Pr[\langle \mathbf{a}_i, \mathbf{s}' \rangle = c_i + u] = \Pr[\langle \mathbf{a}_i, \mathbf{s}' \rangle = \langle \mathbf{a}_i, \mathbf{s}' \rangle + e_i + u] = \Pr[-e_i = u] = p_u.$$

The expected value for $S_{\mathbf{v}}$ in the case of a correct guess is then given by:

$$\mathrm{E}_c := \mathrm{E}(S_{\mathbf{v}} \mid \mathbf{v} = \mathbf{s}') = \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} \Pr[e_i = j] \cdot W(j) = \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} p_j w_j.$$

Finally, for all $j, 1 \leq j \leq \lfloor q/2 \rfloor$, we have $p_{-j} w_{-j} = p_j w_j$. Thus:

$$\sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} p_j w_j = p_0 w_0 + 2 \cdot \sum_{j=1}^{j=\lfloor q/2 \rfloor} p_j w_j.$$

□

We now examine $\mathrm{E}_w = \mathrm{E}(S_{\mathbf{v}} \mid \mathbf{v} \neq \mathbf{s}')$. To begin with, we fix a wrong guess $\mathbf{v}$, such that $\mathbf{v} = \mathbf{s}' + \mathbf{t}$ with $\mathbf{t} \neq 0$.

Lemma 7. *Let* $(\mathbf{a}_0, c_0), \ldots, (\mathbf{a}_{m-1}, c_{m-1})$ *be samples following* $B^{(n)}_{\mathbf{s},\chi,a}$, $\mathbf{v} \in \mathbb{Z}_q^d$, $\tilde{p}_j := \Pr(e \leftarrow_{\$} \mathcal{U}_a : e = j)$, $p_j := \Pr(e \leftarrow_{\$} \chi_a : e = j)$, $w_j := W(j)$, *and* $S_{\mathbf{v}} = \frac{1}{m}\sum_{i=0}^{m-1} W(\langle \mathbf{a}_i, \mathbf{v}\rangle - c_i)$. *If* $\mathbf{v} \neq \mathbf{s}'$, *we have:*

$$\mathrm{E}_w = \mathrm{E}(S_{\mathbf{v}} \mid \mathbf{v} \neq \mathbf{s}') = \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} \tilde{p}_j w_j = \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} \frac{q^{d-1} - p_j}{q^d - 1} w_j. \tag{6}$$

Since the proof of Lemma 7 is analogous to Lemma 6 we omit it here. We now look at the variances Var_c and Var_w.

Lemma 8. *Let* $(\mathbf{a}_0, c_0), \ldots, (\mathbf{a}_{m-1}, c_{m-1})$ *be samples following* $B^{(n)}_{\mathbf{s},\chi,a}$, $\mathbf{v} \in \mathbb{Z}_q^d$, $p_j := \Pr(e \leftarrow_{\$} \chi_a : e = j)$, $\tilde{p}_j := \Pr(e \leftarrow_{\$} \mathcal{U}_a : e = j)$, $w_j := W(j)$ *and* $S_{\mathbf{v}} = \sum_{i=0}^{m-1} \frac{1}{m} W(\langle \mathbf{a}_i, \mathbf{v}\rangle - c_i)$.

If $\mathbf{v} = \mathbf{s}'$, *then*

$$\mathrm{Var}_c := \mathrm{Var}(S_{\mathbf{v}} \mid \mathbf{v} = \mathbf{s}') = \frac{1}{m} \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} p_j \cdot (w_j - \mathrm{E}_c)^2. \tag{7}$$

If $\mathbf{v} \neq \mathbf{s}'$, *then*

$$\mathrm{Var}_w := \mathrm{Var}(S_{\mathbf{v}} \mid \mathbf{v} \neq \mathbf{s}') = \frac{1}{m} \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} \tilde{p}_j \cdot (w_j - \mathrm{E}_w)^2. \tag{8}$$

Proof. In the case of $\mathbf{v} = \mathbf{s}'$ we have that for $m = 1$,

$$\mathrm{Var}_c = \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} p_j \cdot \big(w_j - \mathrm{E}_c\big)^2.$$

In the case of adding then normalising m samples we can use the fact that when adding random variables of zero covariance, the sum of the variances is the variance of the sum. Thus the variance in the case of adding m samples and normalising is given by:

$$\mathrm{Var}_c = m \cdot \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} p_j \cdot \left(\frac{w_j}{m} - \frac{\mathrm{E}_c}{m}\right)^2 = \frac{1}{m} \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} p_j \cdot (w_j - \mathrm{E}_c)^2$$

A similar argument holds in the case of Var_w. □

Finally, given E_c, E_w, Var_c, and Var_w, we can estimate the rank of the right key in dependence of the number of samples m considered. We denote by Y_h the random variable determined by the rank of a correct score $S_{\mathbf{s}'}$ in a list of h elements. Now, for a list of length q^d and a given rank $0 \leq r < q^d$, the probability of Y_{q^d} taking rank r is given by a binomial-normal compound distribution. Finally, we get Lemma 9, which essentially states that for whatever score the right key gets, in order for it to have rank zero the remaining $q^d - 1$ keys must have smaller scores.

Lemma 9. *Let* E_c, E_w, Var_c *and* Var_w *be as in Lemmas 6, 7 and 8. Let also* Y_{q^d} *be the random variable determined by the rank of a correct score* $S_{\mathbf{s}'}$ *in the list* S *of* q^d *elements. Then, the number of samples* m *required for* Y_{q^d} *to take rank zero with probability* $\mathrm{p}_{success}$ *is recovered by solving*

$$\mathrm{p}_{success} = \int_x \left[\frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x - \mathrm{E}_w}{\sqrt{2\mathrm{Var}_w}}\right)\right)\right]^{(q^d-1)} \cdot \left(\frac{1}{\sqrt{2\pi\mathrm{Var}_c}} e^{-\frac{(x-\mathrm{E}_c)^2}{2\mathrm{Var}_c}}\right) \mathrm{d}x,$$

for m.

Proof. Y_{q^d} follows a binomial-normal compound distribution given by $\Pr[Y_{q^d} = r] =$

$$\int_x \left(\binom{q^d - 1}{r} \cdot \Pr[e \leftarrow_\$ D_w : e \geq x]^r \cdot \Pr[e \leftarrow_\$ D_w : e < x]^{(q^d-r-1)} \cdot \Pr[e \leftarrow_\$ D_c : e = x]\right) \mathrm{d}x.$$

Plugging in $r = 0$ and $\Pr[Y_{q^d} = r] = \mathrm{p}_{success}$ we get:

$$\begin{aligned}\mathrm{p}_{success} &= \int_x \Pr[e \leftarrow_\$ D_w : e < x]^{(q^d-1)} \cdot \Pr[e \leftarrow_\$ D_c : e = x] \,\mathrm{d}x \\ &= \int_x \left[\frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x - \mathrm{E}_w}{\sqrt{2\mathrm{Var}_w}}\right)\right)\right]^{(q^d-1)} \cdot \left(\frac{1}{\sqrt{2\pi\mathrm{Var}_c}} e^{-\frac{(x-\mathrm{E}_c)^2}{2\mathrm{Var}_c}}\right) \mathrm{d}x\end{aligned}$$

as required. □

Using Lemma 9 we can hence estimate the number of non-zero samples m we need to recover subvector $\mathbf{s}'$. Finally, we extend this result to recover $\mathbf{s}$.

Remark 2. We note that Algorithm 2 not only returns an ordering of the hypotheses but also a score for each hypothesis. Hence, we can simply sample from $B^{(n)}_{\mathbf{s},\chi,a}$ until the distance between the first and second highest rated hypothesis is above a certain threshold.

3.3 Back Substitution

Given a candidate solution for $\mathbf{s}'$ which is correct with very high probability we can perform backsubstitution in our tables T^i similarly to solving a triangular linear system. It is easy to see that backsubstitution costs $2d$ operations per row. Furthermore, by Lemma 2 we have $a \cdot (\lceil q^b/2 \rceil)$ rows in all tables T^i.

After backsubstitution, we start the BKW algorithm again in stage one where all the tables T^i are already filled. To recover the next d components of $\mathbf{s}$ then, we ask for m samples which are reduced using our modified tables T^i and perform hypothesis testing on these m samples.

3.4 BKW: Complexity

We can now state our main theorem.

Theorem 2. *Let* $(\mathbf{a}_i, c_i)$ *be samples following* $L^{(n)}_{\mathbf{s},\chi}$, $0 < b \leq n$, $d \leq b$ *parameters and* $0 < \mathrm{P}_{success} < 1$ *the targeted success rate. Let* $a = \lceil n/b \rceil$ *and* m *be as in Lemma 9 when* $\mathrm{p}_{success} = (\mathrm{P}_{success})^{\lfloor \frac{d}{n} \rfloor}$. *Then, the expected cost of the BKW algorithm to recover* $\mathbf{s}$ *with success probability* $\mathrm{P}_{success}$ *is*

$$\left(\frac{q^b - 1}{2}\right) \cdot \left(\frac{a(a-1)}{2} \cdot (n+1) - \frac{ba(a-1)}{4} - \frac{b}{6}\left((a-1)^3 + \frac{3}{2}(a-1)^2 + \frac{1}{2}(a-1)\right)\right) \tag{9}$$

additions/subtractions in $\mathbb{Z}_q$ to produce the elimination tables,

$$\frac{q^d}{q^d-1}\cdot\frac{\lceil\frac{n}{d}\rceil+1}{2}\cdot m\cdot\left(\frac{a}{2}\cdot(n+2)\right) \tag{10}$$

additions/subtractions in $\mathbb{Z}_q$ to produce samples for hypothesis testing. For the hypothesis-testing step

$$\left\lceil\frac{n}{d}\right\rceil\cdot\left(m\cdot q^d\right) \tag{11}$$

arithmetic operations in $\mathbb{Z}_q$ are required and

$$\left(\left\lceil\frac{n}{d}\right\rceil+1\right)\cdot d\cdot a\cdot\left\lceil\frac{q^b}{2}\right\rceil \tag{12}$$

operations in $\mathbb{Z}_q$ for backsubstitution. Furthermore,

$$a\cdot(\lceil q^b/2\rceil)+\frac{q^d}{q^d-1}\cdot\left\lceil\frac{n}{d}\right\rceil\cdot m \tag{13}$$

calls to $L^{(n)}_{\mathbf{s},\chi}$ are needed.

Proof. In order to recover $\mathbf{s}$ we need every run of stage 1 to be successful, hence we have $\mathrm{P}_{success} = (\mathrm{p}_{success})^{\lceil n/d\rceil}$ and consequently $\mathrm{p}_{success} = (\mathrm{P}_{success})^{\lfloor\frac{d}{n}\rfloor}$.

Furthermore, we have:

- The cost of constructing the tables T^i in Equation (9) follows from Lemma 2.
- Lemma 2 and the fact that with probability $\frac{1}{q^d}$ the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ returns an all-zero sample establish that to produce m non-zero samples for hypothesis testing, $\frac{q^d}{q^d-1}\cdot m\cdot\left(\frac{a}{2}\cdot(n+2)\right)$ operations are necessary. We need to produce m such samples $\lceil\frac{n}{d}\rceil$ times. However, as we proceed the number of required operations linearly approaches zero. Hence, we need $\frac{\lceil\frac{n}{d}\rceil+1}{2}\cdot\frac{q^d}{q^d-1}\cdot m\cdot\left(\frac{a}{2}\cdot(n+2)\right)$ operations as in Equation (10).
- The cost of Algorithm 2 in Equation (11) which also is run $\lceil\frac{n}{d}\rceil$ times follows from Lemma 4.
- There are $a\cdot\left\lceil\frac{q^b}{2}\right\rceil$ rows in all tables T^i each of which requires $2d$ operations in backsubstitution. We need to run backsubstitution $\lceil\frac{n}{d}\rceil$ times, but each time the cost decreases linearly. From this follows Equation (12).
- The number of samples needed in Equation (13) follows from Lemma 2 and that with probability $\frac{1}{q^d-1}$ the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ returns a sample which is useless to us. □

We would like to express the complexity of the BKW algorithm as a function of n, q, α explicitly. In that regard, Theorem 2 does not deliver. However, from the fact that we can distinguish χ_a and $\mathcal{U}_a$ in subexponential time if the standard deviation of $\chi_a < q/2$, we can derive the following simple corollary eliminating m.

Corollary 1. *Let $(\mathbf{a}_i, c_i)$ be samples following $L^{(n)}_{\mathbf{s},\chi}$, set $a = \lfloor\log_2(1/(2\alpha)^2)\rceil$, $b = n/a$. Let d be a small constant $0 < d < \log_2(n)$. Assume α is such that $q^b = q^{n/a} = q^{n/\log_2(1/(2\alpha)^2)}$ is superpolynomial in n. Then, given these parameters, the cost of the BKW algorithm to recover $\mathbf{s}$ is*

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

operations in $\mathbb{Z}_q$*. Furthermore,*

$$a \cdot (\lceil q^b/2 \rceil) + \text{poly}(n)$$

calls to $L^{(n)}_{\mathbf{s},\chi}$ *are needed.*

Proof (Sketch). From $\chi_a = \sqrt{2^a} \cdot \alpha \cdot q < q/2$ follows $a = \log_2(1/(2\alpha)^2)$. Under this conditions $m \leq q^b$ and Theorem 2 is dominated by terms involving q^b. □

To conclude this section, we discuss under which conditions the BKW algorithm runs in subexponential time. That is, we give conditions on the parameter α of the LWE problem under which the BKW algorithm runs in subexponential time.

Corollary 2. *Let* $(\mathbf{a}_i, c_i)$ *be samples following* $L^{(n)}_{\mathbf{s},\chi}$*, set* $a = \log_2(1/(2\alpha)^2)$*,* $b = n/a$*. The BKW algorithm runs in subexponential time if*

$$\alpha = \mathcal{O}\left(1/q^{F(n)}\right)$$

for any function $F(n)$ *going to infinity with* n*.*

Proof. By Corollary 1, the complexity of the BKW algorithm under these conditions is $\mathcal{O}\left(q^b \cdot \text{poly}(n)\right)$. Our task is to find α – if any – such that

$$q^b = 2^{\log_2(q)\cdot b} \in 2^{o(n)}.$$

By definition, $2^{\log_2(q)\cdot b} \in 2^{o(n)}$ if:

$$\lim_{n\to\infty} \frac{\log_2(q) \cdot b}{n} = 0.$$

We have:

$$\frac{\log_2(q) \cdot b}{n} = \frac{\log_2(q)}{a} = \frac{\log_2(q)}{\log_2(1/(2\alpha)^2)}.$$

Let $F(n)$ be a function going to infinity with n. For

$$\lim_{n\to\infty} \frac{\log_2(q)}{\log_2(1/(2\alpha)^2)} = 0.$$

We have that α must take the form $1/(2\alpha) = q^{F(n)}$. This yields:

$$\lim_{n\to\infty} \frac{\log_2(q)}{a} = \frac{1}{2F(n)} = 0.$$

Hence, the BKW algorithm is subexponential when $\alpha = \mathcal{O}\left(1/q^{F(n)}\right)$ for any function $F(n)$ going to infinity with n. □

Remark 3. This is consistent with existing results on the hardness of the LWE problem [37, 5].

4 BKW on Short-Secret LWE

In [4] a linear-algebra-based method was proposed to transform any LWE instance into an instance where the secret itself can be viewed as having been generated according to $\chi_{\alpha,q}$ at the cost of $\mathcal{O}\left(n^2\right)$ per sample. The

same technique was also applied for LPN in [27]. Furthermore, recent proposals of homomorphic encryption schemes rely on the hardness of LWE instances with very sparse secrets sampled from $\{-1, 0, 1\}$ [20]. Hence, it is a natural question whether the secret vector following $\chi_{\alpha,q}$ as opposed to $\mathcal{U}(\mathbb{Z}_q^n)$ can be exploited to improve the BKW algorithm.

We consider the general case, i.e., that $\mathbf{s}$ is distributed according to $\chi_{\alpha,q}$. In this case, we may employ a 'cut-off', disregarding the 'extremities' of the multivariate discretised Gaussian distribution from which the error vector is derived. The question is then whether by neglecting the hypothesis-testing of a proportion of possible candidates for the error vector, we can offset the additional costs incurred through the transformation from plain LWE to short-secret LWE.

We set a parameter $\xi > 0$, $\xi \in \mathbb{R}$ and assume that for each element e_i, $(0 \leq i \leq d-1)$ of the error vector we have $|e_i| < \beta\xi$, in other words e_i lies within ξ standard deviations. For the typical LWE parameters considered in this work, a reasonable estimate for the probability that a particular element of the error vector has absolute value less than $\xi\beta$ (when we view elements of $\mathbb{Z}_q$ in the centralised representation as integers) is given by $\Pr[|X| < \xi\beta] \approx \operatorname{erf}(\xi/\sqrt{2})$. Thus the probability that a length-d error vector satisfies these conditions is approximately $\gamma = \operatorname{erf}(\xi/\sqrt{2})^d$. Thus, if we restrict our hypothesis-testing stage to error vectors of length-d in which we assume that each element e_i satisfies $|e_i| \leq \beta\xi$, then we reduce the number of candidates to be tested from q^d to $(\lceil 2\beta\xi + 1 \rceil)^d$. This modifies Lemma 2 (we add the transformation cost, cf., Lemma 10), Lemma 4 (there are less candidates to consider, cf., Lemma 11), Lemma 5 ($\tilde{p}_j$ changes), Lemma 7 ($\tilde{p}_j$ changed) and Lemma 9 (the exponent changes, cf., Lemma 12) and hence Theorem 2. The most dramatic modification is due to a reduction of the exponent in Lemma 9.

First, we consider the short secret variant of Lemma 2. We assume that $\lceil 2\xi\beta + 1 \rceil < q$ and that the conditions on a particular error vector of section 4 hold. Then, there is an increased cost for producing samples.

Lemma 10. *(Short-Secret Analogue of Lemma 2) Let $n \geq 1$ be the number of variables, q be a modulus, and $d, b \in \mathbb{Z}$ with $1 \leq d \leq b \leq n$, and define $a = \lceil n/b \rceil$. Let $\overset{s\,(n)}{B}_{\mathbf{s},\chi,n}$ denote an analogous oracle to $B^{(n)}_{\mathbf{s},\chi,n}$, returning short-secret LWE samples. With good probability, the cost of calling $\overset{s\,(n)}{B}_{\mathbf{s},\chi,n}$, which returns samples $(\mathbf{a}_i, c_i)$ with at most the d rightmost entries of $\mathbf{a}_i$ non-zero, m times is upper bounded by:*

$$\left(m + a\,\lceil \frac{q^b}{2} \rceil\right) \cdot (2n+1) + 2n^3 + 2n^2$$

operations in $\mathbb{Z}_q$ for the short-secret conversion stage and

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

additions in $\mathbb{Z}_q$ for the reduction stage.

Proof. The inversion of the initial sample matrix costs $2n^3$ operations in $\mathbb{Z}_q$, and we precompute $\mathbf{A}_0^{-1}\mathbf{c}_0 = \tilde{\mathbf{c}}_0$ while the product $\mathbf{A}_1 \cdot \tilde{\mathbf{c}}_0 - \mathbf{c}_1$ costs $2n^2 + n$ arithmetic operations in $\mathbb{Z}_q$ which is amortised to $2n+1$ operations in $\mathbb{Z}_q$ per short-secret sample of which we need $m + a \cdot (\lceil q^b/2 \rceil)$. □

On the other hand, hypothesis testing is cheaper and the success probability increases. Below, we give analogues of Lemmas 4 and 9. However, we omit their trivial proofs.

Lemma 11. *(Short-Secret Analogue of Lemma 4) For short-secret LWE, running algorithm 2 costs $m \cdot (\lceil 2\beta\xi + 1 \rceil)^d$ operations in $\mathbb{Z}_q$.*

Lemma 12. *(Short-Secret Analogue of Lemma 9) Let* $\mathrm{E}_{(c,s)}$, $\mathrm{E}_{(w,s)}$, $\mathrm{Var}_{(c,s)}$ *and* $\mathrm{Var}_{(w,s)}$ *be the expectation of a correct counter value, expectation of a wrong counter value, variance of a correct counter value and variance of wrong counter value, respectively, when using short-secret LWE. Let* $Y_{\lceil 2\beta\xi+1\rceil^d}$ *be the random variable determined by the rank of a correct score* $S_{\mathbf{s}'}$ *in the list* S *of* $\lceil 2\beta\xi+1\rceil^d$ *elements. Then the number of samples* m *required for* $Y_{\lceil 2\beta\xi+1\rceil^d}$ *to take rank zero with probability* $\mathrm{p}_{success}$ *is recovered by solving*

$$\mathrm{p}_{success} = \int_x \left[\frac{1}{2}\left(1+\mathrm{erf}\left(\frac{x-\mathrm{E}_{(w,s)}}{\sqrt{2\mathrm{Var}_{(w,s)}}}\right)\right)\right]^{(\lceil 2\xi\beta+1\rceil^d-1)} \cdot \left(\frac{1}{\sqrt{2\pi\mathrm{Var}_{(c,s)}}} e^{-\frac{(x-\mathrm{E}_{(c,s)})^2}{2\mathrm{Var}_{(c,s)}}}\right) \mathrm{d}x,$$

for m.

Using these lemmas one can then construct an analogue of Theorem 2. However, when establishing the success probability of this analogous theorem we have to multiply the sucess probability of Theorem 2 by the probability of the target secret $\mathbf{s}$ to lie within ξ standard deviations.

However, neither the construction of the samples m nor the second stage of the algorithm dominate the overall runnning time. Indeed, in our experiments applying this improvement did not result in a significantly lowered complexity but did increase the required number of operations due to the transformation overhead. Of course, this does not rule out that more efficient attacks exist if more substantial modifications are made to the BKW algorithm. We consider combining the BKW algorithm with ideas from [1] and [27] an interesting topic for future work.

5 Selecting Parameters & Applications

In this section we apply Theorem 2 to various sets of parameters suggested in the literature. In order to compute concrete costs we rely on numerical approximations in various places such as the computation of p_j. We used $2n-4n$ bits of precision for all computations which is necessary for these numerical approximations, while increasing this precision further did not appear to change our results. The solving step for m of Lemma 9 is accomplished by a simple search implemented in Sage [40]. As a subroutine of this search we rely on numerical integration which we performed using the mpmath library [26] as shipped with Sage. The Sage script used to derive all values in this section is attached to this document and available at [2].

In all cases below we always set $\mathrm{P}_{success} = 0.99$ and $a := \lfloor t \cdot \log_2 n \rfloor$ where t is a small constant, which is consistent with the complexity of the BKW algorithm $2^{\mathcal{O}(n)} = q^{\mathcal{O}(n/\log_2(n))}$ if $q \in \mathrm{poly}(n)$. Furthermore, we assume that one operation in $\mathbb{Z}_q$ costs $\log_2^2(q)$ bit operations.

We note that we allow an unbounded number of queries to $L^{(n)}_{\mathbf{s},\chi}$, because we are interested in the hardness of the LWE problem per se. While this assumption does not necessarily carry over to all cryptosystems considered in this section, it is folklore [38] that given roughly $n \log q$ LWE samples one can produces many more LWE samples at the cost of an increase in the noise through inter-addition. While employing these approaches would render our proofs inapplicable, it is assumed that in practice similar results would still hold. Similar notions (in the case of LPN) were considered in [17], although, as in this work, the authors did not analyse the impacts of these steps.

5.1 Regev's original parameters

In [37] Regev proposes a simple public-key encryption scheme with the suggested parameters $q \approx n^2$ and $\alpha = 1/(\sqrt{n} \cdot \log_2^2 n)$. We consider the parameters in the range $n = 40, \ldots, 256$. In our experiments $t = 2.6$

produced the best results, i.e., higher values of t resulted in m growing too fast. Plugging these values into the formulas of Theorem 2 we get an overall complexity of

$$\frac{mn^9 + \frac{1}{6}2^{(\frac{2}{2.6}n)}n^5 + \left[\left(2.25\overline{333}\,n + 3.38\right) \cdot \left(2^{\left(\frac{2}{2.6}n\right)}n^4 + 1\right)\right] \cdot \log_2(n)^2 + \frac{1}{6}\,n}{2\,(n^4 - 1)}$$

after simplification. If $m < 2^{(\frac{2}{2.6}n)})$ then this expression is dominated by

$$\frac{\frac{1}{6}n^5 + \left(2.25\overline{333}\,n^5 + 3.38n^4\right) \cdot \log_2(n)^2}{2\,(n^4 - 1)} 2^{\left(\frac{2}{2.6}n\right)} \in 2^{\frac{2}{2.6}n + \mathcal{O}(\log(n))}.$$

However, since we compute m numerically, we have to rely on experimental evidence to verify that with these settings indeed m does not grow too fast. Table 1 lists the estimated number of calls to $L^{(n)}_{\mathbf{s},\chi}$ ("$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$"), the estimated number of required ring ("$\log_2 \#\mathbb{Z}_q$") and bit ("$\log_2 \#\mathbb{Z}_2$") operations, the costs in terms of ring operations for each of the three stages sampling, hypothesis testing and back substitution.

n	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$ in				$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$
		sample	hypo.	subs.	total		
40	31.29	43.21	56.90	38.95	56.90	63.72	35.92
48	33.97	48.00	60.90	45.44	60.90	67.86	40.30
64	39.97	60.48	68.97	58.24	68.97	76.14	52.21
80	45.64	73.27	76.26	70.96	76.47	83.79	64.60
96	51.32	85.94	83.25	83.56	86.37	93.81	76.95
128	62.00	111.16	96.00	108.69	111.40	119.01	101.66
136	65.00	117.43	99.44	114.93	117.66	125.31	107.82
144	67.00	123.70	101.85	121.18	123.93	131.61	113.99
160	73.08	136.21	108.69	133.67	136.44	144.19	126.33
192	84.05	161.20	120.97	158.60	161.42	169.26	151.00
200	86.37	167.43	123.59	164.82	167.65	175.52	157.16
224	94.01	186.12	132.05	183.47	186.33	194.26	175.65
240	99.66	198.56	138.19	195.90	198.77	206.74	187.98
256	105.30	211.00	144.30	208.31	211.21	219.21	200.30

Table 1. Cost of finding $\mathbf{s}$ for parameters suggested in [37] with $d = 2, t = 2.6, \mathrm{P}_{success} = 0.99$.

5.2 Micciancio and Regev's parameters

In their overview of lattice-based cryptosystems [31], Micciancio and Regev discuss how to choose LWE parameters and give concrete candidates for such. Table 2 lists concrete costs for solving LWE under the parameter choices from [31] using the BKW algorithm. These parameters were also considered in [28].

5.3 Albrecht et al.'s PollyCracker

In [3] a somewhat homomorphic encryption scheme is proposed based on the hardness of computing Gröbner bases with noise. Using linearisation the equation systems considered in [3] may be considered as LWE instances. Table 3 lists concrete costs for recovering the secret Gröbner basis using this strategy for selected parameters suggested in [3]. In Table 3 "λ" is the targeted bit-security level and n the number of variables in the linearised system. We note that we did not exploit the structure of the secret for Table 3.

n	q	α	t	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$ in				$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$
					sample	hypo.	subs.	total		
136	2003	0.0065	2.1	78.29	112.38	106.31	110.22	112.69	119.60	103.12
166	4093	0.0024	2.4	75.40	125.50	105.78	123.07	125.75	132.92	115.68
192	8191	0.0009959	2.7	88.69	135.49	121.28	132.83	135.70	143.11	125.23
214	16381	0.00045	3.0	119.59	143.16	154.33	140.29	154.33	161.95	132.56
233	32749	0.000217	3.2	107.30	153.38	144.16	150.40	153.56	161.37	142.53

Table 2. Cost of finding $\mathbf{s}$ for parameters suggested in [31] with $d = 2, \mathrm{P}_{success} = 0.99$.

λ	n	q	α	t	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$ in				$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$
						sample	hypo.	subs.	total		
80	136	1999	0.005582542...	2.2	93.58	109.40	121.59	105.71	121.59	128.50	100.23
	231	92893	0.000139563...	3.4	127.23	157.47	167.09	154.40	167.09	175.18	146.54
128	153	12227	0.002797408...	2.4	84.05	132.07	117.45	129.66	132.32	139.85	122.39
	253	594397	0.000034967...	3.8	100.66	175.15	146.00	171.88	175.29	183.82	163.89

Table 3. Cost of finding $G \approx \mathbf{s}$ for parameters suggested in [3] with $d = 2, \mathrm{P}_{success} = 0.99$.

6 Experimental Results

In order to verify the results of this work, we implemented stages 1 and 2 of the BKW algorithm. Our implementation considers LWE with short secrets but we ignore the transformation cost to produce samples with a short secret. Also, our implementation supports arbitrary bit-width windows b, not only multiplies of $\lceil \log_2(q) \rceil$. However, due to the fact that our implementation does not use a balanced representation of finite field elements internally – which simplifies dealing with arbitrary bit-width windows – our implementation does not *fully* implement the half-table improvement. That is, for simplicity, our implementation only uses the additive inverse of a vector if this is trivially compatible with our internal data representation. Furthermore, our implementation does not bit-pack finite field elements. Elements always take up 16 bits of storage. Overall, the memory consumption of our implementation in stage 1 is worse by a factor of up to four compared to the estimates given in this work and the computational work in stage is worse by a factor of up to two. Finally, since our implementation is not optimised we do not report CPU times.

With these considerations in mind, our estimates are confirmed by our implementation. For example, consider Regev's parameters for $n = 25$ and $t = 2.3$ and $d = 1$, By Lemma 9 picking $m = 2^{12.82}$ will result in a success probability of $\mathrm{p}_{success} \approx 0.99959$ per component and $\mathrm{P}_{success} \approx 0.99$ overall. Lemma 2 estimates a computational cost of $2^{30.54}$ ring operation and $2^{24.19}$ calls to $L^{(n)}_{\mathbf{s},\chi}$ in stage 1. We ran our implementation with $m = \lceil 2^{12.82} \rfloor$ and window bitsize $w = 22 = \frac{n \log_2(q)}{2.3 \log_2(n)}$. It required $2^{29.74}$ ring operations and $2^{23.31}$ calls to $L^{(n)}_{\mathbf{s},\chi}$ to recover one component of $\mathbf{s}$. From this we conclude that Theorem 2 is reasonably tight.

To test the accuracy of Lemma 9 we ran our implementation with the parameters $n = 25$, $q = 631$, $\alpha \cdot q = 5.85$, $w = 24 = \frac{n \log_2(q)}{2.1 \log_2(n)}$ and $m = 2^7$. Lemma 9, adapted to the small secret case (cf. Section 4), predicts a success rate of 53%. In 1000 experiments we 665 times rank zero for the correct key component, while Lemma 9 predicted 530. Hence, it seems our predictions are slightly pessimistic. The distribution of the ranks of the correct component of $\mathbf{s}$ in 1000 experiments is plotted in Figure 1.

Our implementation is available at `http://bitbucket.org/malb/bkw-lwe`.

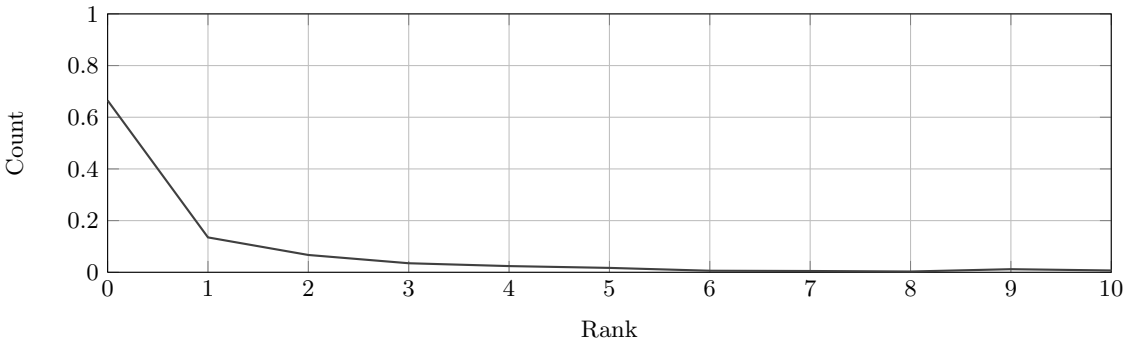

Fig. 1. Distribution of right key component ranks for 1000 experiments on $n = 25$, $t = 2.3$, $d = 1$, $\mathrm{p}_{success} = 0.99$.

7 Comparison with Alternative Approaches

We briefly describe some alternative strategies for solving the LWE problem. However, this section is not intended to be a rigorous comparison, rather an outline of alternative methods. We briefly examine three lattice-based approaches: one closest vector algorithm, an algorithm proposed in [28] which combines lattice basis reduction and enumeration techniques and a distinguishing approach mentioned in [31]. Finally, we consider the recent algorithm of Arora and Ge [5].

7.1 Distinguishing Approach

In [31], the authors briefly examine an approach for solving LWE by distinguishing between valid matrix-LWE samples of the form $(\mathbf{A}_i, \mathbf{c}_i) = (\mathbf{A}, \mathbf{As} + \mathbf{e}_i)$ and samples drawn from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. Given a matrix of samples $\mathbf{A}$, one way of constructing such a distinguisher is to find a short vector $\mathbf{u}$ in the dual lattice $\Lambda(\mathbf{A})^\perp$ such that $\mathbf{uA} = \mathbf{0} \mod q$. If $\mathbf{c}_i$ belongs to the uniform distribution over $\mathbb{Z}_q^n$, then $\langle \mathbf{u}, \mathbf{c}_i \rangle$ belongs to the uniform distribution on $\mathbb{Z}_q$. On the other hand, if $\mathbf{c}_i = \mathbf{As} + \mathbf{e}_i$, then $\langle \mathbf{u}, \mathbf{c}_i \rangle = \langle \mathbf{u}, \mathbf{As} + \mathbf{e}_i \rangle = \langle \mathbf{u}, \mathbf{e}_i \rangle$, where samples of the form $\langle \mathbf{u}, \mathbf{e}_i \rangle$ are governed by another discrete, wrapped Gaussian distribution. Following the work of Micciancio and Regev [31], the authors of [28] give estimates for the complexity of distinguishing between LWE samples and uniform samples by estimating the cost of the BKZ algorithm in finding a short vector satisfying certain conditions. [28] estimates running times of the BKZ algorithm on a 2.3GHz CPU by extrapolating running times from smaller cases. Table 4 gives comparable (conservative) estimates of the number of bit operations, obtained by multiplying the clock speed by the extrapolated running times given by [28].

n	q	α	t	$\log_2 m$	BKW		[28] distinguishing results	
					$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L_{\mathbf{s},\chi}^{(n)}$	$\log_2(\mathrm{s})$	$\log_2(\#\mathbb{Z}_2)$
136	2003	0.0065	2.1	78.29	119.60	103.12	219	250.1
214	16381	0.00045	3.0	119.59	161.95	132.56	126	157.1

Table 4. Cost of distinguishing LWE samples from uniform as reported as "Distinguish" in [28], compared to the cost of recovering $\mathbf{s}$ using BKW.

Furthermore, we may consider the BKW algorithm as presented here as a combinatorial approach for sampling sparse $\mathbf{u}$ with entries $\in \{-1, 0, 1\}$. Hence, it belongs to the same family as, e.g., the combinatorial

approach for finding short vectors, i.e., where each component is in $\in \{-b', \ldots, b'\}$, in q-ary lattices as briefly sketched in [31, p. 156] (cf. also [29]). Note that the parameter b' is unrelated to the parameter b used in this work. The algorithm sketched by [31] uses the generalised birthday paradox to produce collisions among samples produced by inter-addition. That is, expressed in our notation, [31] instantiates a LWE oracle which produces samples where the standard deviation of the error is $\sqrt{b' \cdot \frac{m}{2^k}} \cdot \alpha q$, with k be such that $n \approx \frac{m(k+1)}{2^k} \log_q(2b'+1)$. Then one runs a BKW-style algorithm with

$$b = \frac{m}{2^k} \cdot \log_q(2b'+1)$$

where b' is a parameter of the algorithm depending on $m, \alpha q$ and n. For example, for $n = 80$, $m = n^8$ and Regev's choices for q and α we may pick $b' = 2 \cdot 10^{-11}$ and $k = 10$. This allows to generate samples with a standard deviation of ≈ 3290 on the noise; produced in at least $(2b'+1)^{m/2^k} \approx 2^{94.54}$ operations. This complexity estimate is based on the size of the constructed lists, i.e., $(2b'+1)^{m/2^k}$ in [31] and q^b in our notation. From Theorem 2 it follows that this is indeed a somewhat optimistic lower bound [31] on the actual cost.

7.2 Bounded Distance Decoding

We can take the approach of viewing LWE as being the problem of solving BDD in a random q-ary lattice with a random target $\mathbf{t} = \mathbf{As} + \mathbf{e}$. To solve the LWE problem formulated thus, we have several choices of lattice-based algorithms. However, for several such algorithms, tight complexity estimates are unavailable, thus making comparison to BKW loose. One algorithm for which more precise complexity estimates are available is the AKS (Ajtai, Kumar & Sivakumar) algorithm for solving the shortest vector problem [1]. This algorithm is notable as being the first proposed singly-exponential SVP algorithm.

It is beyond the scope of this paper to give details of the AKS algorithm – we are interested solely in the complexity. In the original paper by Ajtai, Kumar & Sivakumar, no explicit running-time bound was given. Subsequent developments of the AKS algorithm by Regev, Nguyen and Vidick, Micciancio and Voulgaris, Pujol and Stehle delivered algorithms which were of time complexity $2^{16n+o(n)}$, $2^{5.9n+o(n)}$, $2^{3.4n+o(n)}$ and $2^{2.7n+o(n)}$, respectively [23].

7.3 Combination of Lattice Reduction and Decoding Techniques

In [28], the authors propose a method to solve LWE instances which consists of q-ary lattice reduction, then employing a decoding stage to determine the secret.The decoding stage used is essentially a straightforward modification of Babai's well-known nearest plane algorithm for CVP. The authors estimate the running-time of the BKZ algorithm in producing a basis 'reduced-enough' for the decoding stage of the algorithm to succeed, then add the cost of the decoding stage. Several assumptions are both explicit and implicit in this work.

To obtain comparable complexity results, we calculate upper bounds on the bit operation counts for two data-points based on the running times reported in [28] multiplied by the clock speed of the CPU used. As can be seen from Table 5, these indicate substantially lower complexities than for BKW. In addition, the memory requirements of this approach are small compared to the memory requirements of BKW.

7.4 Arora-Ge Algorithm

Recently, Arora and Ge [5] proposed a new approach for solving LWE. The idea is to generate a non-linear noise-free system of equations from LWE samples. This is due to the well-known fact that elements sampled

n	q	α	t	$\log_2 m$	BKW		[28] results	
					$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$	$\log_2(\mathrm{s})$	$\log_2(\#\mathbb{Z}_2)$
136	2003	0.0065	2.1	78.29	119.60	103.12	68	99.1
214	16381	0.00045	3.0	119.59	161.95	132.56	52	83.1

Table 5. Cost of distinguishing LWE samples from uniform as reported as "Decode" in [28], compared to the cost of recovering $\mathbf{s}$ using BKW

from a Gaussian distribution only take values on a (small) subset of $\mathbb{Z}_q$ with high probability. This allows to generate from each LWE sample a degree-D equation in the n components of the secret $\mathbf{s}$. Note that the secret vanishes this equation with a certain probability which can be controlled by the degree D.

The idea of Arora and Ge is then to generate sufficiently many equations to perform a linearization. However, one has to choose the degree D sufficiently high to ensure that the secret is with high probability a common solution of the equations generated. According to [5], taking $D \in \tilde{\mathcal{O}}(\sigma)$ allows to make the probability of failure negligible (recall that σ is the standard deviation of the Gaussian considered). This leads to:

Theorem 3. *[5] Let $D < q$. The system obtained by linearizing $M = \mathcal{O}\left(q \cdot \log(q)\binom{n+D}{D}\sigma\right) = n^{\mathcal{O}(D)} = 2^{\tilde{\mathcal{O}}(D)}$ non-linear degree-D equations generated from* LWE *samples has at most one solution with high probability. The time complexity of the basic Arora-Ge approach is then*

$$n^{\mathcal{O}(D)} = 2^{\tilde{\mathcal{O}}(\sigma^2)}.$$

Note that $\mathcal{O}\left(\binom{n+D}{D}\right)$ equations are sufficient to linearize the system. The extra factor $q \cdot \log(q)\sigma$ allows to prove that the linearized system has at most one solution. The complexity of the approach is then the cost of performing a Gaussian elimination on a matrix of size $M \times \binom{n+D}{D}$.

So far, the main relevance of Arora-Ge's algorithm is from an asymptotical point of view. It allows to prove that LWE can be solved in sub-exponential time when the Gaussian is sufficiently narrow, i.e. $\sigma < \sqrt{n}$. However, as illustrated in Table 6, it is not yet very practical. We have considered the parameters $q \approx n^2$ and $\alpha = 1/(\sqrt{n} \cdot \log_2^2 n)$ as suggested by [37] in the range $n = 40, \ldots, 256$. We list the estimated number of calls to $L^{(n)}_{\mathbf{s},\chi}$ ("$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$"), and the estimated number of required ring ("$\log_2 \#\mathbb{Z}_q$") operations. In our experiments, we have chosen the degree of the equations so that the success probability is ≈ 0.99. We have used $\omega = 3$ for the linear algebra constant.

8 Conclusion and Further Work

In this work we have provided a concrete analysis of the cost of running the BKW algorithm on LWE instances and applied this analysis to various sets of parameters found in the literature. Providing concrete costs allows for a better understanding of both the theoretic aspects of the BKW algorithm as well as the security provided by LWE-based cryptographic schemes. This work focuses on analysing the BKW algorithm with only minimal adaptions to the LWE case. However, further refinements to the BKW algorithm seem possible such as a dedicated variant for small secrets, possibly based on on techniques from [1]. Furthermore, rigorously comparing the results with lattice-based solutions and the Arora-Ge algorithm is necessary to establish the hardness of LWE instances in practice. Finally, combining the BKW algorithm with other techniques such as Arora-Ge is a logical next step.

n	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$	$\log_2 \#\mathbb{Z}_q$
40	215.26	628.05
48	262.16	767.87
64	359.64	1058.98
80	460.88	1361.65
96	565.29	1674.03
128	781.97	2322.68
136	837.51	2489.03
144	893.55	2656.87
160	1006.99	2996.68
192	1238.88	3691.48
256	1719.14	5130.88

Table 6. Cost of finding $\mathbf{s}$ for parameters suggested in [37] with Arora-Ge's algorithm [5].

Acknowledgements

We are grateful to Frederik Johansson for advice on numerical integration. We are also grateful to anonymous referees whose feedback substantially improved this work. The work described in this paper has been partially supported by the Royal Society grant JP090728 and by the Commission of the European Communities through the ICT program under contract ICT-2007-216676 (ECRYPT-II). Jean-Charles Faugère, and Ludovic Perret are also supported by the Computer Algebra and Cryptography (CAC) project (ANR-09-JCJCJ-0064-01) and the HPAC grant of the French National Research Agency.

References

1. Miklós Ajtai, Ravi Kumar, and D. Sivakumar. Sampling short lattice vectors and the closest lattice vector problem. In *IEEE Conference on Computational Complexity*, pages 53–57, 2002.
2. Martin R. Albrecht. `bkw-estimator.py`, 2012. available at `https://bitbucket.org/malb/research-snippets/`.
3. Martin R. Albrecht, Pooya Farshim, Jean-Charles Faugère, and Ludovic Perret. Polly Cracker, revisited. In *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 179–196, Berlin, Heidelberg, New York, 2011. Springer Verlag. full version available as Cryptology ePrint Archive, Report 2011/289, 2011 `http://eprint.iacr.org/`.
4. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology – CRYPTO 2009*, Lecture Notes in Computer Science, pages 595–618, Berlin, Heidelberg, New York, 2009. Springer Verlag.
5. Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415, Berlin, Heidelberg, New York, 2011. Springer Verlag.
6. Thomas Baigneres, Pascal Junod, and Serge Vaudenay. How far can we go beyond Linear Cryptanalysis? In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 432–450, Berlin, Heidelberg, New York, 2004. Springer Verlag.
7. Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard Knapsacks. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 364–385. Springer Verlag, 2011.
8. Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves Information Set Decoding. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer Verlag, 2012.
9. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. In Johannes Buchmann and Jintai Ding, editors, *PQCrypto*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer Verlag, 2008.
10. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011*, volume 6841, pages 743–760. Springer Verlag, 2011.
11. Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid Approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, 3(3):177–197, 2010.
12. Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Solving polynomial systems over finite fields: Improved analysis of the Hybrid Approach. In *ISSAC '12: Proceedings of the 2012 international symposium on Symbolic and algebraic computation*, ISSAC '12, pages 1–12, New York, NY, USA, 2012. ACM.
13. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.
14. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pages 97–106. IEEE, 2011.
15. Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: better lattice security estimates. In *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20, Berlin, Heidelberg, 2011. Springer Verlag.
16. Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105, Berlin, Heidelberg, New York, 2009. Springer Verlag.
17. Pierre-Alain Fouque and Éric Levieil. An improved LPN algorithm. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks, 5th International Conference, SCN 2006*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer Verlag, 2006.
18. Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 257–278. Springer Verlag, 2010.
19. Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. Available at `http://crypto.stanford.edu/craig`.

20. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.
21. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC 08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.
22. Frank Gray. Pulse code communication, March 1953. US Patent No. 2,632,058.
23. Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Algorithms for the shortest and closest lattice vector problems. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *IWCC*, volume 6639 of *Lecture Notes in Computer Science*, pages 159–190. Springer, 2011.
24. Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 447–464. Springer Verlag, 2011.
25. Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard Knapsacks. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 235–256. Springer Verlag, 2010.
26. Fredrik Johansson et al. *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 0.17)*, February 2011. `http://code.google.com/p/mpmath/`.
27. Paul Kirchner. Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377, 2011. `http://eprint.iacr.org/`.
28. Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339, Berlin, Heidelberg, New York, 2011. Springer Verlag.
29. Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. Swifft: A modest proposal for fft hashing. In Kaisa Nyberg, editor, *Fast Software Encryption*, volume 5086 of *Lecture Notes in Computer Science*, pages 54–72. Springer Verlag, 2008.
30. Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124. Springer Verlag, 2011.
31. Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer Verlag, Berlin, Heidelberg, New York, 2009.
32. Ivan Morel, Damien Stehlé, and Gilles Villard. H-LLL: using householder inside LLL. In Jeremy R. Johnson, Hyungju Park, and Erich Kaltofen, editors, *Symbolic and Algebraic Computation, International Symposium, ISSAC 2009*, pages 271–278. ACM, 2009.
33. Phong Q. Nguyen. Lattice reduction algorithms: Theory and practice. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 2–6. Springer Verlag, 2011.
34. Phong Q. Nguyen and Damien Stehlé. Low-dimensional lattice basis reduction revisited. *ACM Transactions on Algorithms*, 5(4), 2009.
35. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 333–342. ACM, 2009.
36. Xavier Pujol and Damien Stehlé. Solving the shortest lattice vector problem in time $2^{2.465n}$. *IACR Cryptology ePrint Archive*, 2009:605, 2009.
37. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.
38. Oded Regev. The learning with errors problem (invited survey). In *IEEE Conference on Computational Complexity*, pages 191–204. IEEE Computer Society, 2010.
39. Markus Rückert and Michael Schneider. Estimating the security of lattice-based cryptosystems. *IACR Cryptology ePrint Archive*, 2010:137, 2010.
40. W. A. Stein et al. *Sage Mathematics Software (Version 5.2)*. The Sage Development Team, 2012. `http://www.sagemath.org`.