# Lazy Modulus Switching for the BKW Algorithm on LWE

Martin R. Albrecht[1], Jean-Charles Faugère[3,2,4], Robert Fitzpatrick[5], and Ludovic Perret[2,3,4]

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

**Abstract.** Some recent constructions based on LWE do not sample the secret uniformly at random but rather from some distribution which produces small entries. The most prominent of these is the binary-LWE problem where the secret vector is sampled from $\{0,1\}^*$ or $\{-1,0,1\}^*$. We present a variant of the BKW algorithm for binary-LWE and other small secret variants and show that this variant reduces the complexity for solving binary-LWE. We also give estimates for the cost of solving binary-LWE instances in this setting and demonstrate the advantage of this BKW variant over standard BKW and lattice reduction techniques applied to the SIS problem. Our variant can be seen as a combination of the BKW algorithm with a lazy variant of modulus switching which might be of independent interest.

## 1 Introduction

Learning With Errors (LWE) [20] has received widespread attention from the cryptographic community since its introduction. LWE-based cryptography is mainly motivated by its great flexibility for instantiating cryptographic solution as well as a deep worst-case/average-case connections [20]: solving LWE on the average is not easier than solving worst-case instances of several famous lattice approximation problems.

The motivation behind this work comes from the observation that some recent constructions based on LWE do not sample the secret uniformly at random but rather from some distribution which produces small entries (e.g. [5,1,13,12,19]). From a theoretical point of view, this is motivated by the observation that every LWE instance can be transformed into an instance where the secret follows the same distribution as the noise [5].[1] However, many constructions use secrets

[1] Also in [15] for the LPN case.

H. Krawczyk (Ed.): PKC 2014, LNCS 8383, pp. 429–445, 2014.
© International Association for Cryptologic Research 2014

which are considerably smaller. For example, binary-LWE samples the secret from $\{0,1\}^*$ [8] or $\{-1,0,1\}^*$ [12]. The presence of such small secrets provokes the question of what implications such choices have on the security of LWE. Is solving LWE with, say, binary secrets easier than standard LWE? From a theoretical point of view, [8] proves that their binary-LWE is as secure as LWE. In this paper, we try to address the question from an algorithmic point of view; i.e. what is the actual impact of small secrets on concrete parameters.

### 1.1 Algorithms for Solving LWE

Three families of algorithms for solving LWE are known in the literature. The most prominent approach is to reduce LWE to a problem that can be solved via lattice reduction, for example, by reducing it to the Short Integer Solution (SIS) problem. Indeed, most parameter choices in the literature are based on the hardness of lattice reduction such as [16,10,17]. These estimates for a given set of parameters $n$ (number of components of the secret), $q$ (size of the modulus) and $\sigma$ (standard deviation of the noise) are usually produced by extrapolating running times from small instances.

A second approach is due to Arora and Ge who reduce LWE to solving a system of non-linear equations [6]. This algorithm allow us to solve LWE in sub-exponential time as soon as the Gaussian distribution is sufficiently narrow, i.e. $\alpha \cdot q < \sqrt{n}$. Recall that the security reduction [20] for LWE requires to consider discrete Gaussian with standard deviation $\alpha \cdot q$ strictly bigger than $\sqrt{n}$. However, from a practical point of view, the constants involved in this algorithm are so large that it is much more costly than other approaches for the parameters typically considered in cryptographic applications [2].

The third family of algorithms are combinatorial algorithms which can all be seen as variants of the BKW algorithm. The BKW algorithm was proposed by Blum, Kalai and Wasserman [7] as a method for solving the Learning Parity with Noise problem, with sub-exponential complexity, requiring $2^{\mathcal{O}(n/\log n)}$ samples, space and time. The algorithm can be adapted for tackling LWE with complexity $2^{\mathcal{O}(n)}$ when the modulus is taken to be polynomial in $n$ [20]. BKW proceeds by splitting the $n$ components of LWE samples into $a$ groups of $b$ components each. For each of the $a$ groups of components the algorithm then searches for collisions in these $b$ components to eliminate them. The overall complexity of the algorithm is $\approx \left(a^2 n\right) \cdot \frac{q^b}{2}$ operations, and $a \cdot \frac{q^b}{2}$ memory, where $a$ and $b$ depend on the $n, q$ and $\alpha$.

The behaviour of the algorithm is relatively well understood and it was shown to outperform lattice reduction estimates when reducing LWE to SIS (when $q$ is small), thus it provides a solid basis for analysing the concrete hardness of LWE instances [3].

### 1.2 Organisation of the Paper and Main Results

While none of the algorithms above take advantage of the presence of small secrets, we may combine them with *modulus switching*. Recall that modulus

switching was initially introduced to improve the performance of homomorphic encryption schemes [9] and was recently used to reduce the hardness of LWE with polynomially sized moduli to GAPSVP [8]. Modulus switching is essentially the same as computing with a lower precision similar to performing floating point computations with a low fixed precision. Namely, let $\big(\mathbf{a}, c = \langle \mathbf{a}, \mathbf{s}\rangle + e\big) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ be LWE sample where $\mathbf{s} \in \mathbb{Z}_q^n$ is the secret vector, and $e \in \mathbb{Z}_q$ is an error. Let also some $p < q$ and consider $\big(\lfloor p/q \cdot \mathbf{a}\rceil, \lfloor p/q \cdot c\rceil\big)$ with

$$\begin{aligned}
\left\lfloor \frac{p}{q} \cdot c \right\rceil &= \left\lfloor \frac{p}{q}\big(\langle \mathbf{a}, \mathbf{s}\rangle + q \cdot u + e\big) \right\rceil, \text{ for some } u \in \mathbb{Z} \\
\left\lfloor \frac{p}{q} \cdot c \right\rceil &= \left\lfloor \left\langle \frac{p}{q} \cdot \mathbf{a}, \mathbf{s} \right\rangle_p + \frac{p}{q} \cdot e \right\rceil = \left\lfloor \left\langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p + \left\langle \frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p + \frac{p}{q} \cdot e \right\rceil \\
&= \left\langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p + \left\langle \frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p + \frac{p}{q} \cdot e + e', \text{ where } e' \in [-0.5, 0.5] \\
&= \left\langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p + e'' + \frac{p}{q} \cdot e + e'. \qquad (1)
\end{aligned}$$

where $\langle \mathbf{x}, \mathbf{y}\rangle_p$ denotes the modulo $p$ inner product of $\mathbf{x}$ and $\mathbf{y}$.

Since $p/q \cdot \mathbf{a} - \lfloor p/q \cdot \mathbf{a}\rceil$ takes values $\in [-0.5, 0.5]$ we have that $e''$ is small if $\mathbf{s}$ is small. We may hence compute with the smaller 'precision' $p$ at the cost of a slight increase of the noise rate by a '*rounding error*' $e''$.

Modulus switching allows to map a LWE instance $\bmod q$ to a scaled instance of LWE $\bmod p$. Thus, modulus switching can be used in the solving of small secret instances of LWE, a folklore approach which has not been explicitly studied in the literature. Namely, if we pick $p$ such that $e''$ is not much larger than $p/q \cdot e$ then, for example, the running time of the BKW algorithm improves from $(a^2 n) \cdot \frac{q^b}{2}$ to $(a^2 n) \cdot \frac{p^b}{2}$. Since typically $b \approx n/\log n$ this may translate to substantial improvements. Indeed, we can pick $p$ such that $|\langle p/q \cdot \mathbf{a} - \lfloor p/q \cdot \mathbf{a}\rceil, \mathbf{s}\rangle| \approx p/q \cdot |e|$. This implies $\sigma_s \cdot \sqrt{\frac{n}{12}} \approx p/q \cdot \sigma$, or $p \approx \min\big\{q, \frac{\sigma_s}{\sigma} \cdot \sqrt{\frac{n}{12}} \cdot q\big\}$, where $\sigma_s$ is the standard deviation of elements in the secret $\mathbf{s}$.

In this paper, we refine this approach and present a variant of the BKW algorithm which fuses modulus switching and BKW-style reduction. In particular, this work has two main contributions. Firstly, in Section 2 we present a modulus switching strategy for the BKW algorithm in which switching is delayed until necessary. In a nutshell, recall that the BKW algorithm performs additions of elements which collide in certain components. Our variant will search for such collisions in 'low precision' $\mathbb{Z}_p$ but will perform arithmetic in 'high precision' $\mathbb{Z}_q$. We call *rounding error* the inner product of the sub-vector of 'low bits' of $\mathbf{a}$ with the secret $\mathbf{s}$. Our strategy permits to decrease rounding errors and allows to reduce $p$ by a factor of $\sqrt{a}$.

Secondly, this perspective enables us to choose reductors in the BKW algorithm which minimise the rounding errors further (Section 3). Namely, we favour components $a$ with small distance $|\lfloor p/q \cdot a\rceil - p/q \cdot a|$ in already reduced components, called 'child components' in this work. Our strategy ensures that the probability of finding such elements is highest for those components which are

considered first by the BKW algorithm, i.e. those components which contribute most to the noise. We note that the first contribution relies on standard independence assumptions only, while the second contribution relies on stronger assumptions, which however seem to hold in practice.

We then discuss the complexity of our variants in Section 4. For typical choices of parameters – i.e. $q \approx n^c$ for some small constant $c \geq 1$, $a = \log_2 n$ and $b = n/\log_2 n$ – the complexity of BKW as analysed in [3] is $\mathcal{O}\left(2^{cn} \cdot n \log_2^2 n\right)$. For small secrets, a naive modulus switching technique allows reducing this complexity to $\mathcal{O}\left(2^{n\left(c+\frac{\log_2 d}{\log_2 n}\right)} \cdot n \log_2^2 n\right)$ where $0 < d \leq 1$ is a small constant. If the secret distribution does not depend on $n$ and if an unbounded number of LWE samples is available our improved version of BKW allows to get a complexity of:

$$\mathcal{O}\left(2^{n\left(c+\frac{\log_2 d-\frac{1}{2}\log_2\log_2 n}{\log_2 n}\right)} \cdot n \log_2^2 n\right).$$

We then study the behaviour of this algorithm by applying it to various instances of LWE with binary secrets. In Section 5, we compare the results with plain BKW and BKZ under modulus switching and a simple meet-in-the-middle approach or generalised birthday attack. We show that our lazy-modulus-switching variant of the BKW algorithm provides better results than applying plain BKW after modulus reduction. We also demonstrate that under the parameters considered here this algorithm also – as $n$ increases – outperforms the most optimistic estimates for BKZ when we apply BKZ to the same task as that to which we apply BKW: finding short vectors in the (scaled-)dual lattice – we obtain this perspective by viewing the rounding error as an increase in the noise rate while still finding short vectors in the (scaled)-dual $p$-ary lattice determined by our modulus-reduced LWE samples. Indeed, our results indicate that our algorithm outperforms BKZ 2.0 when both are used to find a short vector in the (scaled)-dual lattice in dimension as low as $\approx 256$ when considering LWE parameters from [20] with binary secret. However, we stress again that we always assume an unbounded number of samples to be available for solving.

### 1.3 Notations

To fix the notations, we reproduce below the definition of LWE.

**Definition 1 (LWE [20]).** *Let $n$, $q$ be positive integers, $\chi$ be a probability distribution on $\mathbb{Z}_q$ and $\mathbf{s}$ be a secret vector in $\mathbb{Z}_q^n$. We denote by $L_{\mathbf{s},\chi}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \in \mathbb{Z}_q$ according to $\chi$, and returning $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. We define* Decision-LWE *as the problem of deciding whether pairs $(\mathbf{a}, c) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ are sampled according to $L_{\mathbf{s},\chi}$ or the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.* Search-LWE *is the problem of recovering $\mathbf{s}$ from $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ sampled according to $L_{\mathbf{s},\chi}$.*

The noise follows some distribution $\chi$ which is classically chosen to be a discrete Gaussian distribution over $\mathbb{Z}$ with mean 0 and standard deviation

$\sigma = s/\sqrt{2\pi} = \alpha q/\sqrt{2\pi}$, reduced modulo $q$. In the following, we always start counting at zero. We denote vectors as well as matrices in bold, vectors in lower case, and matrices in upper case. 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 entry at index $i, j$. For vectors $\mathbf{a}$ we denote by $\mathbf{a}_{(a,b)}$ the vector $(\mathbf{a}_{(a)}, \ldots, \mathbf{a}_{(b-1)})$. 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. This means that $\mathbf{a}_{i,(j)}$ is the $j$-th component of the vector $\mathbf{a}_i$. 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 initial distribution. We write $\tilde{\mathbf{a}}$ instead of $\mathbf{a}$ to indicate $\mathbf{a}$ has some short elements. We represent elements in $\mathbb{Z}_q$ as integers in $[-\frac{q}{2}, \ldots, \frac{q}{2}]$, similarly for $\mathbb{Z}_p$. We write $\chi_{\alpha,q}$ for the distribution obtained by considering a discrete Gaussian distribution over $\mathbb{Z}$ with standard deviation $\alpha q/\sqrt{2\pi}$, mean 0, considered modulo $q$.

## 2 A Modified BKW Algorithm: Lazy Modulus Switching

Following [3], we consider BKW – applied to Decision-LWE – as consisting of two stages: *sample reduction* and *hypothesis testing*. In this work, we only modify the first stage.

### 2.1 The Basic Idea

We briefly recall the principle of classical BKW. Assume we are given samples of the form $(\mathbf{a}, c)$ following either $L_{\mathbf{s},\chi}$ or $\mathcal{U}(\mathbb{Z}_q^n) \times \mathcal{U}(\mathbb{Z}_q)$. Our goal is to distinguish between the two cases. BKW proceeds by producing samples $(\mathbf{a}^*, c^*)$ with $\mathbf{a}^*$ being all zero such that statistical tests can be applied to $c^*$ to decide whether they follow $\mathcal{U}(\mathbb{Z}_q)$ or some distribution related to $L_{\mathbf{s},\chi}$. This is achieved by grouping the $n$ components of all vectors into $a$ groups of $b$ components each (assuming $a$ and $b$ divide $n$ for simplicity). If two vectors collide on all $b$ entries in one group, the first is subtracted from the second, producing a vector with at least $b$ all zero entries. These vectors are then again combined to produce more all zero entries and so forth until all $a$ groups are eliminated to zero. However, as we add up vectors the noise increases. Overall, after $\ell$ addition levels the noise has standard deviation $\sqrt{2^\ell}\alpha q$. Our algorithm, too, will be parametrized by a positive integer $b \leq n$ (the window width), and $a := \lceil n/b \rceil$ (the addition depth).

Recall that the complexity of BKW algorithm is essentially $q^b$. However, $b$ only depends on the ratio $\alpha q/\sqrt{2\pi}q = \alpha\sqrt{2\pi}$ and thus not on $q$. Hence, it is clear that applying modulus reduction before running the BKW algorithm may greatly improve its running time: $b$ is preserved whilst $q$ is reduced to $p$. However, instead of applying modulus reduction in 'one shot' prior to executing BKW, we propose switching to a lower precision only when needed. For this, we actually never switch the modulus but simply consider elements in $\mathbb{Z}_q$ 'through the perspective' of $\mathbb{Z}_p$. We then essentially only consider the top-most $\log_2 p$ bits of $\mathbb{Z}_q$.

Under this perspective, given samples of the form $(\mathbf{a}, c)$ we aim to produce $\left(\tilde{\mathbf{a}}, \tilde{c} = \langle \tilde{\mathbf{a}}, \mathbf{s} \rangle + \tilde{e}\right)$, where $\tilde{\mathbf{a}}$ is short enough, i.e.

$$|\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle| \approx \sqrt{2^a} \alpha q. \tag{2}$$

Although other choices are possible, this choice means balancing the noise $\tilde{e}$ after $a$ levels of addition and the contribution of $|\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle|$ such that neither dominates. We call the term $\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle$ the *rounding error*. So, condition (2) is such that after $a$ levels of additions performed by the BKW algorithm the escalated initial noise and the noise coming from rounding errors have the same size.

### 2.2 Sample Reduction for Short Secrets

Let $(\mathbf{a}_0, c_0), \ldots, (\mathbf{a}_{m-1}, c_{m-1})$ be samples which follow $L_{\mathbf{s},\chi}$ or $\mathcal{U}(\mathbb{Z}_q^n) \times \mathcal{U}(\mathbb{Z}_q)$. We now explain how to produce samples $(\tilde{\mathbf{a}}_i, \tilde{c}_i)_{i \geq 0}$ that satisfy condition (2). For simplicity, we assume from now on that $p = 2^\kappa$. [2]

The main idea of the algorithm is to search for collisions among the first $b$ components of samples $(\mathbf{a}_i, c_i)$ by only considering their top $\log_2 p$ bits. If such a collision is found, we proceed as in the normal BKW algorithm, i.e. we subtract the colliding samples to clear the first $b$ components. In our case, we clear the top-most $\log_2 p$ bits of the first $b$ components. Hence, instead of managing elimination tables for every bit of all components, we only manage elimination tables for the most significant $\kappa$ bits. Put differently, all arithmetic is performed in $\mathbb{Z}_q$ but collisions are searched for in $\mathbb{Z}_p$ after rescaling or modulus switching.

As in [3], we realise the first stage of the BKW algorithm as a (recursively constructed) series of oracles $B_{\mathbf{s},\chi}(b, \ell, p)$. In our case, we have $0 \leq \ell < a$, where $B_{\mathbf{s},\chi}(b, a-1, p)$ produces the final output and $B_{\mathbf{s},\chi}(b, -1, p)$ calls the LWE oracle. We will make use of a set of tables $T^\ell$ (maintained across oracle calls) to store (randomly-chosen) vectors that will be used to reduce samples arising from our oracles. However, compared to [3] our oracles $B_{\mathbf{s},\chi}(b, \ell, p)$ take an additional parameter $p$ which specifies the precision which we consider. Hence, if $p = q$ then we recover the algorithm from [3] where we perform no modulus reduction at all. In particular, $B_{\mathbf{s},\chi}(b, \ell, p)$ proceeds as follows:

1. For $\ell = -1$, we can obtain samples from $B_{\mathbf{s},\chi}(b, -1, p)$ by simply calling the LWE oracle $L_{\mathbf{s},\chi}$ and returning the output.
2. For $\ell = 0$, we repeatedly query the oracle $B_{\mathbf{s},\chi}(b, 0, p)$ to obtain (at most) $(p^b - 1)/2$ samples $(\mathbf{a}, c)$ with distinct non-zero vectors $\lfloor p/q \cdot \mathbf{a}_{(0,b)} \rceil$. We use these samples to populate the table $T^0$, indexed by $\lfloor p/q \cdot \mathbf{a}_{(0,b)} \rceil$. We store $(\mathbf{a}, c)$ in the table. During this course of this population, whenever we obtain a sample $(\mathbf{a}', c')$ from $B_{\mathbf{s},\chi}(b, -1, p)$, if $\left\lfloor p/q \cdot \mathbf{a}'_{(0,b)} \right\rceil$ (resp. the negation) match $\lfloor p/q \cdot \mathbf{a}_{(0,b)} \rceil$ such that the pair $(\mathbf{a}, c)$ is already in $T^1$, we return $(\mathbf{a}' \pm \mathbf{a}, c' \pm c)$, as a sample from $B_{\mathbf{s},\chi}(b, 0, p)$. Note that, if $\lfloor p/q \cdot \mathbf{a}_{(0,b)} \rceil$

[2] While we do not have to restrict our attention to $p$ of the form $2^\kappa$, we choose it for ease of exposition and implementation.

is zero, we return $(\mathbf{a}', c')$ as a sample from $B_{\mathbf{s},\chi}(b, 0, p)$. Further calls to the oracle $B_{\mathbf{s},\chi}(b, 0, p)$ proceed in a similar manner, but using (and potentially adding entries to) the same table $T^0$.

3. For $0 < \ell < a$, we proceed as above: we make use of the table $T^\ell$ (constructed by calling $B_{\mathbf{s},\chi}(b, \ell - 1, p)$ up to $(p^b - 1)/2$ times) to reduce any output sample from $B_{\mathbf{s},\chi}(b, \ell - 1, p)$ with $\lfloor p/q \cdot \mathbf{a}_{(b\cdot\ell, b\cdot\ell+b)} \rceil$ by an element with a matching such vector, to generate a sample returned by $B_{\mathbf{s},\chi}(b, \ell, p)$.

Pseudo-code for the modified oracle $B_{\mathbf{s},\chi}(b, \ell, p)$, for $0 \leq \ell < a$, is given in the full version of this work.

### 2.3 Picking $p$

Yet, we still have to establish the size of $p$ to satisfy Condition 2. We note that in our approach we do not actually multiply by $p/q$. Let $\sigma_r$ be the standard deviation of uniformly random elements in $\mathbb{Z}_{\lfloor q/p \rceil}$. Performing one-shot modulus switching in this setting would mean splitting $\mathbf{a}$ into two vectors, $\mathbf{a}'$ with the 'high order' bits and $\mathbf{a}''$ with 'low order' bits. The components of the latter would contribute to the final noise as the rounding error, the components of the former would be eliminated by BKW. The standard deviation of the components of $\mathbf{a}''$ is $\sigma_r$. For each component of $\mathbf{a}_{(i)}$ one-shot modulus switching would add a noise with standard deviation $\sigma_r \sigma_s$. Hence, after applying BKW to these pre-processed samples, the standard deviation of the noise contributed by modulus-switching in the final output would be

$$\sqrt{n \cdot 2^a \cdot \sigma_r^2 \sigma_s^2} = \sqrt{a\, b \cdot 2^a \cdot \sigma_r^2 \sigma_s^2}. \tag{3}$$

However, as the following lemma establishes, we may consider smaller $p$ because the final noise contributed by modulus switching in our algorithm is smaller than in (3). This is because if $(\tilde{\mathbf{a}}_i, \tilde{c}_i)$ are final output samples then the entries $\tilde{\mathbf{a}}_{i,(b\cdot a-1)}$ will be significantly smaller than $\tilde{\mathbf{a}}_{i,(0)}$.

Yet, to formalise this, we need to make a (standard) simplifying assumption, namely that the outputs of the BKW algorithm (at every stage) are independent. That is, we make the assumption that, during the course of the algorithm described, all components of each sample from $B_{\mathbf{s},\chi}(b, \ell, p)$ are independent from every other sample. We emphasize that similar assumptions are standard in treatments of combinatorial algorithms for LPN/LWE (cf. [3,11]).

**Assumption 1.** *We assume that all outputs of $B_{\mathbf{s},\chi}(b, \ell, p)$ are independent.*

Assumption 1 allows to establish the following lemma:

**Lemma 1.** *Let $n \geq 1$ be the dimension of the* LWE *secret vector, $q$ be a modulus, $b \in \mathbb{Z}$ with $1 \leq b \leq n$. Let also $\sigma_r$ be the standard deviation of uniformly random elements in $\mathbb{Z}_{\lfloor q/p \rceil}$. Under Assumption 1, the components of $\tilde{\mathbf{a}} = \mathbf{a} - \mathbf{a}'$ returned by $B_{\mathbf{s},\chi}(b, \ell, p)$ satisfy:*

$$\mathrm{Var}(\tilde{\mathbf{a}}_{(i)}) = 2^{\ell - \lfloor i/b \rfloor} \sigma_r^2, \text{ for } 0 \leq \lfloor i/b \rfloor \leq \ell$$

*and* $\mathrm{Var}\big(\mathcal{U}(\mathbb{Z}_q)\big)$ *for* $\lfloor i/b \rfloor > \ell$.

*Proof.* The proof is omitted here but available in the full version of this work.

Using Lemma 1 we may adapt our choice of $p$, because the noise contributed by modulus switching for a given $p$ is smaller:

**Corollary 1.** *Let $n \geq 1$ be the dimension of the* LWE *secret vector, $q$ be a modulus, $b \in \mathbb{Z}$ with $1 \leq b \leq n$. Let $\sigma_r$ be the standard deviation of uniformly random elements in $\mathbb{Z}\lfloor q/p \rceil$ and $\sigma_s$ be the standard deviation of the distribution from which the secret $\mathbf{s}$ is sampled. Let $(\tilde{a}, \tilde{c})$ be an output of $B_{\mathbf{s},\chi}(b, a-1, p)$. Under Assumption 1, the noise added by lazy modulus switching in the final output of $B_{\mathbf{s},\chi}(b, a-1, p)$, that is $|\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle|$, has standard deviation*

$$\sqrt{b \cdot \left( \sum_{i=0}^{a-1} 2^{a-i-1} \right) \cdot \sigma_r^2 \sigma_s^2} = \sqrt{b \cdot (2^a - 1) \cdot \sigma_r^2 \sigma_s^2}.$$

*Proof.* The proof is omitted here but available in the full version of this work.

Now, compare Corollary 1 with the standard deviation in (3). We see that the standard deviation obtained using our lazy modulus switching is divided by a factor $\sqrt{a}$ w.r.t. to a naive use of modulus-switching, i.e. as in (3). As a consequence, we may reduce $p$ by a factor $\sqrt{a}$.

## 3 Improved Algorithm: Stunting Growth by Unnatural Selection

Based on the strategy in the previous section, we now introduce a pre-processing step which allows us to further reduce the magnitude of the noise present in the outputs of $B_{\mathbf{s},\chi}(b, a-1, p)$ by reducing rounding errors further. For this, it will be useful to establish notation to refer to various components of $\mathbf{a}_i$ in relation to $B_{\mathbf{s},\chi}(b, \ell, p)$.

**Children:** are all those components with index $j < b \cdot \ell$, i.e. those components that were reduced by some $B_{\mathbf{s},\chi}(b, k, p)$ with $k < \ell$: *they grow up so quickly.*

**Parents:** are those components of $\mathbf{a}_i$ with index $b \cdot \ell \leq j < b \cdot \ell + b$, i.e. those components among which collisions are searched for in $B_{\mathbf{s},\chi}(b, \ell, p)$: *collisions among parents produce children.*

**Strangers:** with respect to $B_{\mathbf{s},\chi}(b, \ell, p)$ are all other components $j \geq b \cdot \ell + b$: *they are indifferent towards each other.*

### 3.1 The Basic Idea

For the general idea and intuition, assume $b = 1$ and that $\tilde{\mathbf{a}}_i$ are outputs of $B_{\mathbf{s},\chi}(b, 0, p)$ and we hence have $\mathrm{Var}(\tilde{\mathbf{a}}_{i,(0)}) = \sigma_r^2$. Now, some of these $\tilde{\mathbf{a}}_i$ will be stored in Table $T^1$ by $B_{\mathbf{s},\chi}(b, 1, p)$ based on the value in the parent component $\tilde{\mathbf{a}}_{i,(1)}$. All future outputs of $B_{\mathbf{s},\chi}(b, 1, p)$ which collide with $\tilde{\mathbf{a}}_i$ in the parent component at index 1 will have $\tilde{\mathbf{a}}_i$ added/subtracted to it, we are hence adding a value with $\mathrm{Var}(\tilde{\mathbf{a}}_{i,(0)}) = \sigma_r^2$ in index 0.

Now, however, if the $\tilde{\mathbf{a}}_{i,(0)}$ happened to be unusually short, all $B_{\mathbf{s},\chi}(b,\ell,p)$ for $\ell > 0$ would output vectors with a shorter $\tilde{\mathbf{a}}_{i,(0)}$ added/subtracted in, i.e. would also have unusually small child components (although to a lesser degree). That is, improving the outputs of $B_{\mathbf{s},\chi}(b,1,p)$ – i.e. decreasing the magnitude of the $\tilde{\mathbf{a}}_{i,(0)}$ stored in $T^1$ – has a knock-on effect on all later outputs. More generally, improving the outputs of $B_{\mathbf{s},\chi}(b,\ell,p)$ will improve the outputs of $B_{\mathbf{s},\chi}(b,k,p)$ for $k > \ell$.

On the other hand, improving the outputs of $B_{\mathbf{s},\chi}(b,\ell,p)$ where $\ell$ is small, is easier than for larger values of $\ell$. In the algorithm as described so far, when we obtain a collision between a member of $T^\ell$ and an output $(\mathbf{a}_i, c_i)$ of $B_{\mathbf{s},\chi}(b,\ell-1,p)$, we reduce $(\mathbf{a}_i, c_i)$ using the colliding member of $T^\ell$, retaining this member in the table. Alternatively we can reduce $(\mathbf{a}_i, c_i)$ using the *in-situ* table entry, replace the table entry with (the now reduced) $(\mathbf{a}_i, c_i)$ and return the former table entry as the output of $B_{\mathbf{s},\chi}(b,\ell,p)$. If we selectively employ this alternative strategy using the relative magnitudes of the child components of $(\mathbf{a}_i, c_i)$ and the table entry as a criterion, we can improve the 'quality' of our tables as part of a pre-processing phase.

That is, in $B_{\mathbf{s},\chi}(b,\ell,p)$ for each collision in a parent component we may inspect the child components for their size and keep that in $T^\ell$ where the child components are smallest. Phrased in the language of 'children' and 'parents': we do not let 'nature', i.e. randomness, run its course but intervene and select children based on their size. As the number of child components is $b \cdot \ell$ it becomes more difficult as $\ell$ increases to find vectors where all child components are short.

### 3.2 Algorithms

This leads to a modified algorithm $B_{small,\mathbf{s},\chi}(b,\ell,p)$ given in Algorithm 1 which acts as a pre-processing phase.

**begin**
  $T^\ell \leftarrow$ table with $p^b$ rows maintained across all runs of $B_{small,\mathbf{s},\chi}(b,\ell,p)$;
  Find $(\mathbf{a}', c') \leftarrow T^\ell_{\mathbf{z}}$ that collides with a fresh sample $(\mathbf{a}, c)$ from $B_{\mathbf{s},\chi}(b,\ell-1,p)$;
  **if** $\sum_{i=0}^{b\cdot\ell-1} \left|\mathbf{a}'_{(i)}\right| > \sum_{i=0}^{b\cdot\ell-1} \left|\mathbf{a}_{(i)}\right|$ **then**
    $T^\ell_{\mathbf{z}} \leftarrow (\mathbf{a}, c)$;
  **return** $(\mathbf{a}-\mathbf{a}', c-c')$;

**Algorithm 1.** $B_{small,\mathbf{s},\chi}(b,\ell,p)$ for $0 \le \ell < a$

### 3.3 Picking $p$

It remains to be established what the effect of such a strategy is, i.e. how fast children grow up or how fast rounding errors accumulate. In particular, given $n$ vectors $\mathbf{x}_i$ sampled from some distribution $\mathcal{D}$ where each component has standard

deviation $\sigma$, i.e. $\mathrm{Var}(\mathbf{x}_{i,(j)}) = \sigma^2$ we are interested in the standard deviation $\sigma_n$ of each component for $\mathbf{x}^* = \min_{abs}(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1})$ where $\min_{abs}$ picks that vector where $\sum_{j=0}^{b\cdot\ell-1} \left|\mathbf{x}_{(j)}\right|$ is minimal. At this point we know no closed algebraic expression for $\sigma_n$. However, we found (as detailed in the full version of this work) that $\sigma_n$ can be estimated as follows:

**Assumption 2.** *Let the vectors $\mathbf{x}_0, \ldots, \mathbf{x}_{n-1} \in \mathbb{Z}_q^\tau$ be sampled from some distribution $\mathcal{D}$ such that $\sigma^2 = \mathrm{Var}(\mathbf{x}_{i,(j)})$ where $\mathcal{D}$ is any distribution on (sub-)vectors observable in our algorithm. Let $\mathbf{x}^* = \min_{abs}(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1})$ where $\min_{abs}$ picks that vector $\mathbf{x}^*$ with $\sum_{j=0}^{b\cdot\ell-1} \left|\mathbf{x}^*_{(j)}\right|$ minimal. The stddev $\sigma_n = \sqrt{\mathrm{Var}(\mathbf{x}^*_{(0)})} = \cdots = \sqrt{\mathrm{Var}(\mathbf{x}^*_{(\tau-1)})}$ of components in $\mathbf{x}^*$ satisfies*

$$\sigma/\sigma_n \geq c_\tau \sqrt[\tau]{n} + (1 - c_\tau)$$

*with $c_\tau$ as in Table 1 for $\tau \leq 10$ and*

$$c_\tau = 0.20151418166952917\sqrt{\tau} + 0.32362108131969386 \approx \frac{1}{5}\sqrt{\tau} + \frac{1}{3}$$

*otherwise.*

**Table 1.** $c_\tau$ for small values of $\tau$

| $\tau$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $c_\tau$ | 0.405799353869 | 0.692447899282 | 0.789885269135 | 0.844195936036 | 0.854967912468 |
| $\tau$ | 6 | 7 | 8 | 9 | 10 |
| $c_\tau$ | 0.895446987232 | 0.91570933651 | 0.956763578012 | 0.943424544282 | 0.998715322134 |

With Assumption 2 we can now estimate the size of the entries of the variance matrix associated with our elimination tables. That is, a matrix $\mathbf{M}$ where the entry $\mathbf{M}_{(i,j)}$ holds the variance of entries $(b \cdot j, \ldots, b \cdot j + b - 1)$ in $T^i$. We give an algorithm for constructing $\mathbf{M}$ in Algorithm 2 which repeatedly applies Assumptions 1 and 2. We discuss this algorithm in detail and back up the expectation that it gives a reasonable approximation of the variances in $T^\ell$ with empirical evidence the full version of this work.

Using the matrix $\mathbf{M}$ computed by Algorithm 2, we can estimate the variances of components of $\tilde{\mathbf{a}}_i$ as output by $B_{\mathbf{s},\chi}(b, a-1, p)$. This result follows immediately from Assumption 2.

**Lemma 2.** *Let $n \geq 1, q$ be a modulus, $b \in \mathbb{Z}$ with $1 \leq b \leq n$ and $\sigma_r$ be the standard deviation of $\mathcal{U}(\mathbb{Z}_{\lfloor q/p \rceil})$. Define $a := \lceil n/b \rceil$ and pick some $p < q$ and let $\mathbf{M}$ be the output of Algorithm 2 under these parameters. Let $(\tilde{\mathbf{a}}_i, c_i)$ be samples returned by $B_{\mathbf{s},\chi}(b, a-1, p)$. Finally, define $\mathbf{v}$ as the $a-$vector of variances of the components of $\tilde{\mathbf{a}}$ where $\mathbf{v}_{(k)}$ holds the variance of the components $\tilde{\mathbf{a}}_{(b\cdot k)}$ to $\tilde{\mathbf{a}}_{(b\cdot k+b-1)}$. Under Assumption 2, the components of $\mathbf{v}$ satisfy:*

$$\mathbf{v}_{(i)} = \sigma_r^2 + \sum_{j=i+1}^{a} \mathbf{M}_{(j,i)}.$$

```
begin
    T ← 2 · p^b/2; // fudge factor: 2
    n ← m*/((a+1)·T) + 1;
    Var_red = Var(U(Z_⌊q/p⌉)) = σ_r^2; // the var. of fresh red. elements
    M is an a × a matrix;
    for 0 ≤ r < a do
        for 0 ≤ c < a do
            M_(r,c) ← Var(U(Z_q)); // el. on and above main diag. not red.
    for 1 ≤ t < a do
        // row t = sum of prev. rows + 1 fresh el. for each index
        for 0 ≤ i < t do
            M_(t,i) ← Var_red + Σ_{j=i+1}^{t-1} M_(j,i);
        τ ← b · ℓ;
        for 0 ≤ i < t do
            M_(t,i) ← M_(t,i) / (c_τ ∛n + 1 − c_τ)^2;
```

**Algorithm 2.** Constructing $\mathbf{M}$

This now allows us to given an expression for the noise distribution output by $B_{\mathbf{s},\chi}(b, a-1, p)$.

**Lemma 3.** *Let $n \geq 1$ be the dimension of the* LWE *secret vector, $q$ be a modulus, $b \in \mathbb{Z}$ with $1 \leq b \leq n$. Define $a := \lceil n/b \rceil$ and pick some $p < q$ and let $\mathbf{v}$ be as in Lemma 2. Let $(\tilde{\mathbf{a}}_i, \tilde{c}_i)$ be outputs of $B_{\mathbf{s},\chi}(b, a-1, p)$. We assume that Assumptions 1 and 2 hold. Then as $a$ increases the distribution of $\tilde{c}_i$ approaches a discrete Gaussian distribution modulo $q$ with standard deviation*

$$\sigma_{total} := \sqrt{2^a \sigma + b\, \sigma_r^2 \sigma_s^2 \sum_{i=0}^{a-1} \mathbf{v}_{(i)}} \leq \sqrt{2^a \sigma + (2^a - 1) \cdot b \cdot \sigma_r^2 \sigma_s^2}.$$

*Proof.* The standard deviation follows from Assumption 1 and Lemma 2. Since the distribution is formed by adding up $2^a$ vectors it approaches a discrete Gaussian distribution when considered over $\mathbb{Z}$ as $a$ increases by the Central Limit Theorem. □

**Assumption 3.** *We assume that Lemma 3 holds for $128 \leq n$, i.e. the values of $n$ considered in this work.*

## 4 Complexity

Finally, we analyse the complexity of the presented algorithms. To do so, we assume that Assumptions 1, 2, and 3 hold. Lemma 3 allows us to estimate the numbers of samples needed to distinguish the outputs of $B_{\mathbf{s},\chi}(b, a-1, p)$ if $B_{\mathbf{s},\chi}(b, -1, p)$ returns LWE samples from uniform. For this, we rely on standard

estimates [16] for the number of samples required to distinguish. This estimate provides a good approximation for the advantage obtainable in distinguishing between $\mathcal{U}(\mathbb{Z}_q)$ and a discrete Gaussian reduced mod $q$ with standard deviation $\sigma_{total}$. In particular, we compute the advantage as

$$\mathrm{Adv} = \exp\left(-\pi\left(\frac{\sigma_{total}\cdot\sqrt{2\pi}}{q}\right)^2\right).$$

We can now state the overall complexity of running the algorithm in Theorem 1. Remark that the proof of next two results are omitted; they follow by an easy adaptation of the proof of Lemma 2 in [3].

**Theorem 1.** *Let $n \geq 1$ be the dimension of the* LWE *secret vector, $q$ be a modulus, $b \in \mathbb{Z}$ with $1 \leq b \leq n$ and $\sigma_s$ the standard deviation of the secret vector components. Let also $\sigma_r$ be the variance of random elements in $\mathbb{Z}_{\lfloor q/p_{\mathrm{small}}\rceil}$. Define $a := \lceil n/b \rceil$ and pick a pair $(p_{\mathrm{small}}, m^*)$ such that $b\,\sigma_r^2\sigma_s^2 \sum_{i=0}^{a-1} \mathbf{v}_{(i)} \leq 2^a\sigma$, where $\mathbf{v}_{(i)}$ is defined as in Lemma 3. Then $B_{\mathbf{s},\chi}(b, a-1, p)$ will return $(\tilde{\mathbf{a}}_0, \tilde{c}_0), \ldots, (\tilde{\mathbf{a}}_{m-1}, \tilde{c}_{m-1})$ where $\tilde{c}_i$ has standard deviation $\leq \sqrt{2^{a+1}}\cdot\sigma$. Furthermore, this costs*

$$\frac{p_{\mathrm{small}}^b}{2}\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)\right) + (m+m^*)\;n\,a$$

*additions in $\mathbb{Z}_q$ and $a\cdot\left(\frac{p_{\mathrm{small}}^b}{2}\right) + m + m^*$ calls to $L_{\mathbf{s},\chi}$.*

The memory requirement for storing each table is established in Corollary 2 below.

**Corollary 2.** *The memory required to store the table $T^i$ is upper-bounded by*

$$\frac{p_{\mathrm{small}}^b}{2}\cdot a\cdot(n+1)$$

*elements in $\mathbb{Z}_q$, each of which requires $\lceil \log_2(q)\rceil$ bits of storage.*

To clarify the impact of Theorem 1, we consider $m^* = 0$ – i.e. the case discussed in Section 2 – on classical parameters of LWE.

**Corollary 3.** *Let $q \approx n^c$, for some constant $c > 0$, and $\alpha = n^{1/2-c}$ such that $\sigma \approx \alpha q \approx \sqrt{n}$. Furthermore, let $a = \log_2 n$ and $b = n/\log_2 n$ be the usual choices of parameters for BKW. Assume $\sigma_s$ does not depend on $n$. Then, solving Decision-*LWE *costs at most*

$$\mathcal{O}\left(2^{n\left(c+\frac{\log_2 d-\frac{1}{2}\log_2\log_2 n}{\log_2 n}\right)}\cdot n\,\log_2^2 n\right)$$

*operations in $\mathbb{Z}_q$. We also need to store $\mathcal{O}\left(2^{n\left(c+\frac{\log_2 d-\frac{1}{2}\log_2\log_2 n}{\log_2 n}\right)}\cdot n\,\log_2 n\right)$ elements in $\mathbb{Z}_q$.*

*Proof.* The proof is omitted here but available in the full version of this work.

## 5 Parameters

To understand the behaviour of our more careful modulus switching technique for concrete parameters, we compare it with one-shot modulus switching. Specifically, we consider the "plain" BKW algorithm [7] as analysed in [3]. Furthermore, to make this work somewhat self-contained we also compare with the BKZ (2.0) algorithm when applied to SIS instances derived from LWE samples and with a simple meet-in-the-middle (MITM) approach or generalised birthday attack.

Instances. We choose $n \in [128, 256, 512, 1024, 2048]$ and – using [4] – pick $q \approx n^2$ and $\sigma = \frac{q}{\sqrt{2\pi n \log_2^2 n}}$ as in Regev's original encryption scheme [20]. We then consider binary-LWE as defined in [12]: $\mathbf{s} \leftarrow_\$ \mathcal{U}(\{-1, 0, 1\}^n)$ (we consider the case $\mathbf{s} \leftarrow_\$ \mathcal{U}(\mathbb{Z}_2^n)$ as in [8] in the full version of this work). However, we assume an unbounded number of samples being available to the attacker to establish the performance of the algorithms discussed here under optimal conditions.

BKW. For complexity estimates of the plain BKW algorithm we rely on [3]. There the BKW algorithm takes a parameter $t$ which controls the addition depth $a := t \log_2 n$. Here we first pick $t = 2(\log_2 q - \log_2 \sigma)/\log_2 n$ which ensures that the standard deviation of the noise after $a$ levels of additions grows only as large as the modulus. We then slowly increase $t$ in steps of 0.1 until the performance of the algorithm is not estimated to improve any further because too many samples are needed to perform the distinguishing step. Following [3], we translate operations in $\mathbb{Z}_q$ into "bit operations" by multiplying by $\log_2 q$.

BKZ. To estimate the cost of the BKZ (2.0) algorithm we follow [18,16]. In [18], the authors briefly examine an approach for solving LWE by distinguishing between valid matrix-LWE samples of the form $(\mathbf{A}, \mathbf{c}) = (\mathbf{A}, \mathbf{As}+\mathbf{e})$ 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 $\mathbf{u}$ such that $\mathbf{uA} = \mathbf{0} \bmod q$. If $\mathbf{c}$ belongs to the uniform distribution over $\mathbb{Z}_q^n$, then $\langle \mathbf{u}, \mathbf{c} \rangle$ belongs to the uniform distribution on $\mathbb{Z}_q$. On the other hand, if $\mathbf{c} = \mathbf{As} + \mathbf{e}$, then $\langle \mathbf{u}, \mathbf{c} \rangle = \langle \mathbf{u}, \mathbf{As} + \mathbf{e} \rangle = \langle \mathbf{u}, \mathbf{e} \rangle$, 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 [18], the authors of [16] 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 enough vector. In particular, given $n, q, \sigma$ and a target distinguishing advantage $\epsilon$ we set $s = \sigma \cdot \sqrt{2\pi}$ and compute $\beta = q/s \cdot \sqrt{\log(1/\epsilon)/\pi}$. From this $\beta$ we then compute the required root Hermite factor $\delta_0 = 2^{\log_2^2(\beta)/(4n \log_2 q)}$.

Given $\delta_0$ we then approximate the running time of BKZ 2.0 in seconds using two different strategies. Both strategies treat $\delta_0$ as the dominant influence in determining the running time. The first strategy denoted "BKZ" follows [16] and defines $\log_2 T_{sec} = 1.8/\log_2 \delta_0 - 110$. The second strategy denoted "BKZ2" follows [3] who interpolated data points from [17] as $\log_2 T_{sec} = 0.009/\log_2^2 \delta_0 - 27$.

We translate the running time in seconds figure into bit operations by assuming $2.3 \cdot 10^9$ bit operations per second on a 2.3 GHz CPU, which is pessimistic. Furthermore, for BKZ choosing advantage $\epsilon \ll 1$ and running the algorithms about $1/\epsilon$ times is usually more efficient than choosing $\epsilon \approx 1$ directly, i.e. we generate a new lattice of optimal sub-dimension each time using fresh LWE samples.

MITM. One can also solve small secret LWE with a meet-in-the-middle attack that requires $\approx \mathfrak{c}^{n/2}$ time and space where $\mathfrak{c}$ is the cardinality of the set from which each component of the secret is sampled (so $\mathfrak{c} = 2$ or $\mathfrak{c} = 3$ for binary-LWE depending on the definition used): compute and store a sorted list of all $\mathbf{A}\mathbf{s}'$ where $\mathbf{s}' = (\mathbf{s}_{(0)}, \ldots, \mathbf{s}_{(n/2)-1}, 0, 0, \ldots, 0)$ for all possible $\mathfrak{c}^{n/2}$ choices for $\mathbf{s}'$. Then compute $\mathbf{c} - \mathbf{A}\mathbf{s}''$ where we have $\mathbf{s}'' = (0, 0, \ldots, 0, \mathbf{s}_{(n/2)}, \ldots, \mathbf{s}_{n-1})$ and check for a vector that is close to this value in the list.

In Table 2 we give the number of bit operations ("$\log \mathbb{Z}_2$"), calls to the LWE oracle ("$\log L_{\mathbf{s},\chi}$") and memory requirement ("log mem") for BKW without any modulus reduction to establish the baseline. All costs are given for the high advantage case, i.e. if $\epsilon \ll 1$ we multiply the cost by $1/\epsilon$.

Table 3 gives the running times after modulus reduction with $p = q\sqrt{n/12}\sigma_s/\sigma$. In particular, Table 3 lists the expected running time (number of oracle calls and where applicable memory requirements) of running BKW and BKZ after applying modulus reduction.

Finally, Table 4 gives the expected costs for solving these LWE instances using the techniques described in this work. We list two variants: one with and one without "unnatural selection". This is because these techniques rely on more assumptions than the rest of this work which means we have greater confidence in the predictions avoiding such assumptions.

**Table 2.** Cost for solving Decision-LWE with advantage $\approx 1$ for BKW, BKZ and MITM where $q$ and $\sigma$ are chosen as in [20] and $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{-1, 0, 1\}^n)$

| | MITM | | BKZ [16] | | | BKZ 2.0 [17] | | | BKW [3] | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $\log \mathbb{Z}_2$ | log mem | $\log \epsilon$ | $\log L_{\mathbf{s},\chi}$ | $\log \mathbb{Z}_2$ | $\log \epsilon$ | $\log L_{\mathbf{s},\chi}$ | $\log \mathbb{Z}_2$ | $t$ | $\log L_{\mathbf{s},\chi}$ | $\log \mathbb{Z}_2$ | log mem |
| 128 | 105.2 | 101.4 | -18 | 26.5 | 65.4 | -14 | 22.5 | 65.7 | 3.18 | 83.9 | 97.6 | 90.0 |
| 256 | 206.9 | 202.9 | -29 | 38.5 | 179.5 | -35 | 44.5 | 178.5 | 3.13 | 167.2 | 182.1 | 174.2 |
| 512 | 409.9 | 405.8 | -48 | 58.5 | 390.9 | -94 | 104.5 | 522.8 | 3.00 | 344.7 | 361.0 | 352.8 |
| 1024 | 815.8 | 811.5 | -82 | 93.5 | 785.0 | -265 | 276.5 | 1606.2 | 2.99 | 688.0 | 705.5 | 697.0 |
| 2048 | 1627.5 | 1623.0 | -141 | 153.6 | 1523.6 | -773 | 785.4 | 5100.0 | 3.00 | 1369.8 | 1388.7 | 1379.9 |

**Table 3.** Cost for solving Decision-LWE with advantage $\approx 1$ for BKW and BKZ variants where $q, \sigma$ are chosen as in [20] and $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{-1, 0, 1\}^n)$ after one-shot modulus reduction with $p = q\sqrt{n/12}\sigma_s/\sigma$

| | BKZ [16] | | | BKZ 2.0 [17] | | | BKW [3] | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $\log \epsilon$ | $\log L_{\mathbf{s},\chi}$ | $\log \mathbb{Z}_2$ | $\log \epsilon$ | $\log L_{\mathbf{s},\chi}$ | $\log \mathbb{Z}_2$ | $t$ | $\log L_{\mathbf{s},\chi}$ | $\log \mathbb{Z}_2$ | log mem |
| 128 | -21 | 29.3 | 70.2 | -16 | 24.4 | 69.8 | 2.85 | 76.8 | 90.2 | 82.4 |
| 256 | -31 | 40.3 | 175.3 | -37 | 46.3 | 172.8 | 2.85 | 150.4 | 165.6 | 153.7 |
| 512 | -50 | 60.3 | 365.0 | -90 | 100.2 | 467.0 | 2.76 | 293.8 | 309.6 | 301.9 |
| 1024 | -81 | 92.3 | 710.1 | -236 | 247.2 | 1339.1 | 2.78 | 570.3 | 587.4 | 579.4 |
| 2048 | -134 | 146.3 | 1342.3 | -647 | 659.2 | 4006.5 | 2.71 | 1149.0 | 1167.3 | 1159.1 |

**Table 4.** Cost for solving Decision-LWE with advantage $\approx 1$ with the algorithms discussed in this work when $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{-1,0,1\}^n)$

| | this work (w/o unnatural selection) | | | | | | this work | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $t$ | $\log p$ | $\log m^*$ | $\log L_{\mathbf{s},\chi}$ | $\log \mathbb{Z}_2$ | log mem | $t$ | $\log p$ | $\log m^*$ | $\log L_{\mathbf{s},\chi}$ | $\log \mathbb{Z}_2$ | log mem |
| 128 | 2.98 | 10 | 0 | 64.7 | 78.2 | 70.8 | 2.98 | 6 | 61 | 61.0 | 75.2 | 46.3 |
| 256 | 2.83 | 11 | 0 | 127.8 | 142.7 | 134.9 | 2.83 | 5 | 118 | 118.0 | 133.5 | 67.1 |
| 512 | 2.70 | 11 | 0 | 235.1 | 251.2 | 243.1 | 2.70 | 8 | 225 | 225.0 | 241.8 | 180.0 |
| 1024 | 2.59 | 12 | 0 | 477.4 | 494.8 | 486.5 | 2.59 | 10 | 467 | 467.0 | 485.0 | 407.5 |
| 2048 | 2.50 | 12 | 0 | 971.4 | 990.7 | 907.9 | 2.50 | 12 | 961 | 961.0 | 980.2 | 907.9 |

Discussion. The results in this section indicate that the variants of the BKW algorithms discussed in this work compare favourably for the paramters considered. The results in this table also indicate that the unnatural selection strategy has little impact on the overall time complexity. However, it allows to reduce the data complexity, in some cases, considerably. In particular, e.g. considering line 1 of Table 4, we note that applying this technique can make the difference between a feasible ($\approx 80 \cdot 1024^4$ bytes) and infeasible ($\approx 1260 \cdot 1024^6$ bytes) attack for a well-equipped attacker [14]. Finally, we note that our results indicate that lattice reduction benefits from modulus reduction. However, this seems somewhat implausible judging from the used algorithms. This might indicate that the lattice reduction estimates from the literature above might need to be revised.

## 6 Conclusion and Future Work

We investigated applying modulus switching to exploit the presence of a small secret in LWE instances and demonstrated that it can make a significant impact on the complexity of solving such instances. We also adapted the BKW algorithm to perform modulus-switching 'on-the-fly', showing that this approach is superior to performing 'one-shot' modulus reduction on LWE samples prior to solving. Our first variant improves the target modulus by a factor of $\sqrt{\log_2 n}$ in typical scenarios; our second variant mainly improves the memory requirements of the algorithm, one of the key limiting aspects of the BKW algorithm. Our algorithms, however, rely on various assumptions which, though appearing sound, are unproven. Our estimates should thus be considered heuristic, as are performance estimates for all currently-known algorithms for solving LWE. Verifying these assumptions is hence a promising direction for future research. Furthermore, one of the main remaining obstacles for applying the BKW algorithm to cryptographic constructions based on LWE is that it requires an unbounded number of samples to proceed. Lifting this requirement, if only heuristically, is hence a pressing research question.

**Acknowledgement.** We thank Steven Galbraith for helpful comments on an earlier draft of this work. We also thank anonymous referees for detailed comments which greatly improved this work. Jean-Charles Faugère, and Ludovic Perret have been partially supported supported by the Computer Algebra and Cryptography (CAC) project (ANR-09-JCJCJ-0064-01) and the HPAC grant (ANR ANR-11-BS02-013) of the French National Research Agency.

## References


1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
2. Albrecht, M., Cid, C., Faugẗe, J.-C., Fitzpatrick, R., Perret, L.: On the complexity of the arora-ge algorithm against lwe. In: SCC 2012: Proceedings of the 3rd International Conference on Symbolic Computation and Cryptography, Castro-Urdiales, pp. 93–99 (July 2012)
3. Albrecht, M.R., Cid, C., Faugère, J.-C., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. Designs, Codes and Cryptography, 1–30 (2013)
4. Albrecht, M.R., Fitzpatrick, R., Cabracas, D., Gpfert, F., Schneider, M.: A generator for LWE and Ring-LWE instances (2013), `http://www.iacr.org/news/files/2013-04-29lwe-generator.pdf`
5. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)
6. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 403–415. Springer, Heidelberg (2011)
7. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM 50(4), 506–519 (2003)
8. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of Learning with Errors. In: STOC 2013, pp. 575–584. ACM, New York (2013)
9. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, pp. 97–106. IEEE (2011)
10. Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
11. Levieil, É., Fouque, P.-A.: An improved LPN algorithm. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 348–359. Springer, Heidelberg (2006)
12. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
13. Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Robustness of the Learning with Errors assumption. In: ICS, pp. 230–240. Tsinghua University Press (2010)
14. Hill, K.: Blueprints of NSA's ridiculously expensive data center in Utah suggest it holds less info than thought (2013), `http://www.forbes.com/sites/kashmirhill/2013/07/24/blueprints-of-nsa-data-center-in-utah-suggest-its-storage-capacity-is-less-impressive-than-thought/`
15. Kirchner, P.: Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377 (2011), `http://eprint.iacr.org/`
16. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. IACR Cryptology ePrint Archive, 2010:592 (2010)

17. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 293–309. Springer, Heidelberg (2013)
18. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer, Heidelberg (2009)
19. Pietrzak, K.: Subspace LWE. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 548–563. Springer, Heidelberg (2012)
20. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6) (2009)