

Estimation of the Hardness of the Learning with Errors Problem with a Restricted Number of Samples

Nina Bindel · Johannes Buchmann · Florian Göpfert · Markus Schmidt

Received: date / Accepted: date

Abstract The Learning with Errors problem (LWE) is one of the most important hardness assumptions lattice-based constructions base their security on. Recently, Albrecht et al. (Journal of Mathematical Cryptology, 2015) presented the software tool *LWE-Estimator* to estimate the hardness of concrete LWE instances, making the choice of parameters for lattice-based primitives easier and better comparable. To give lower bounds on the hardness it is assumed that each algorithm has given the corresponding optimal number of samples. However, this is not the case for many cryptographic applications.

In this work we first analyze the hardness of LWE instances given a restricted number of samples. For this, we describe LWE solvers from the literature and estimate their runtime considering a limited number of samples. Based on our theoretical results we extend the LWE-Estimator. Furthermore, we evaluate LWE instances proposed for cryptographic schemes and show the impact of restricting the number of available samples.

Keywords lattice-based cryptography · learning with errors problem · LWE · post-quantum cryptography

This work has been supported by the the German Research Foundation (DFG) as part of project P1 within the CRC 1119 CROSSING.

Nina Bindel
Department of Computer Science, Technische Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, Germany,
Tel.: +49-6151-16-20667
Fax: +49-6151-16-6036
E-mail: nbindel@cdc.informatik.tu-darmstadt.de

Johannes Buchmann, Florian Göpfert, Markus Schmidt
Department of Computer Science, Technische Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, Germany

1 Introduction

The Learning with Errors (LWE) problem is used in the construction of many cryptographic lattice-based primitives [18, 28, 30]. It became popular due to its flexibility for instantiating very different cryptographic solutions and its (presumed) hardness against quantum algorithms. Moreover, LWE can be instantiated such that it is provably as hard as worst-case lattice problems [30].

In general, an instance of LWE is characterized by parameters $n \in \mathbb{Z}$, $\alpha \in (0, 1)$, and $q \in \mathbb{Z}$. To solve an instance of LWE, an algorithm has to recover the secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, given access to m LWE samples $(\mathbf{a}_i, c_i = \mathbf{a}_i \cdot \mathbf{s} + e_i \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where the coefficients of \mathbf{s} and the e_i are small and chosen according to probability distribution characterized by α (see Definition 2).

To ease the hardness estimation of concrete instances of LWE, the *LWE-Estimator* [3, 4] was introduced. In particular, the LWE-Estimator is a very useful software tool to choose and compare concrete parameters for lattice-based primitives. To this end, the LWE-Estimator summarizes and combines existing attacks to solve LWE from the literature. The effectiveness of LWE solvers often depend on the number of given LWE samples. To give conservative bounds on the hardness of LWE, the LWE-Estimator assumes that the *optimal* number of samples is given for each algorithm, i.e., the number of samples for which the algorithm runs in minimal time. However, in cryptographic applications the optimal number of samples is often not available. In such cases the hardness of used LWE instances estimated by the LWE-Estimator might be overly conservative. Hence, also the

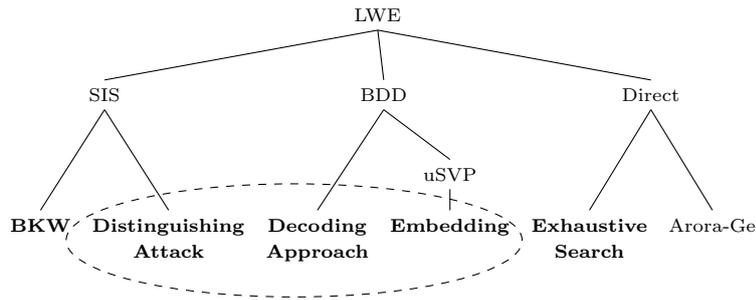


Fig. 1: Overview of existing LWE solvers categorized by different solving strategies described in Section 2; algorithms using basis reduction are dashed-framed; algorithms considered in this work are written in bold

system parameters of cryptographic primitives based on those hardness assumptions are more conservative than necessary from the viewpoint of state-of-the-art cryptanalysis. A more precise hardness estimation is to take the restricted number of samples given by cryptographic applications into account.

In this work we close this gap. We extend the theoretical analysis and the LWE-Estimator such that the hardness of an LWE instance is computed when only a restricted number of samples is given. As Albrecht et al. [4], our analysis is based on the following algorithms: exhaustive search, the Blum-Kalai-Wassermann (BKW) algorithm, the distinguishing attack, the decoding attack, and the standard embedding approach. In contrast to the existing LWE-Estimator we do not adapt the algorithm proposed by Arora and Ge [8] due to its high costs and consequential insignificant practical use. Additionally, we also analyze the dual embedding attack. This variant of the standard embedding approach is very suitable for instances with a small number of samples since the embedding lattice is of dimension $m + n + 1$ instead of $m + 1$ as in the standard embedding approach. Hence, it is very important for our case with restricted number of samples. As in [4], we also analyze and implement small secret variants of all considered LWE solvers, where the coefficients of the secret vector are chosen from a predefined set of small numbers, with a restricted number of samples given.

Moreover, we evaluate our implementation to show that the hardness of most of the considered algorithms are influenced significantly by limiting the number of available samples. Furthermore, we show how the impact of reducing the number of samples differs depending on the model the hardness is estimated in.

Our implementation is already integrated into the existing LWE-Estimator at <https://bitbucket.org/malb/lwe-estimator> (from commit-id eb45a74 on). In our implementation, we always use the existing estimations

with optimal number of samples, if the given restricted number of samples exceeds the optimal number. If not enough samples are given, we calculate the computational costs using the estimations presented in this work.

Figure 1 shows the categorization by strategies used to solve LWE: One approach reduces LWE to finding a short vector in the dual lattice formed by the given samples, also known as Short Integer Solution (SIS) problem. Another strategy solves LWE by considering it as a Bounded Distance Decoding (BDD) problem. The *direct* strategy solves for the secret directly. In Figure 1, we dash-frame algorithms that make use of basis reduction methods. The algorithms considered in this work are written in bold.

In Section 2 we introduce notations and definitions required for the subsequent sections. In Section 3 we describe basis reduction and its runtime estimations. In Section 4 we give our analyses of the considered LWE solvers. In Section 5 we describe and evaluate our implementation. In Section 6 we explain how restricting the number of samples impacts the bit-hardness in different models. In Section 7 we summarize our work.

2 Preliminaries

2.1 Notation

We follow the notation used by Albrecht et al. [4]. Logarithms are base 2 if not indicated otherwise. We write $\ln(\cdot)$ to indicate the use of the natural logarithm. Column vectors are denoted by lowercase bold letters and matrices by uppercase bold letters. Let \mathbf{a} be a vector then we denote the i -th component of \mathbf{a} by $\mathbf{a}_{(i)}$. We write \mathbf{a}_i for the i -th vector of a list of vectors. Moreover, we denote the concatenation of two vectors \mathbf{a} and \mathbf{b} by $\mathbf{a} \parallel \mathbf{b} = (\mathbf{a}_{(1)}, \dots, \mathbf{a}_{(n)}, \mathbf{b}_{(1)}, \dots, \mathbf{b}_{(n)})$ and

$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n \mathbf{a}_{(i)} \mathbf{b}_{(i)}$ is the usual dot product. We denote the euclidian norm of a vector \mathbf{v} with $\|\mathbf{v}\|$.

With $D_{\mathbb{Z}, \alpha q}$ we denote the discrete Gaussian distribution over \mathbb{Z} with mean zero and standard deviation $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$. For a finite set S , we denote sampling the element s uniformly from S with $s \leftarrow_{\S} S$. Let χ be a distribution over \mathbb{Z} , then we write $x \leftarrow \chi$ if x is sampled according to χ . Moreover, we denote sampling each coordinate of a matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$ with distribution χ by $\mathbf{A} \leftarrow \chi^{m \times n}$ with $m, n \in \mathbb{Z}_{>0}$.

2.2 Lattices

For definitions of a lattice L , its rank, its bases, and and its determinant $\det(L)$ we refer to [4]. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, we define the lattices $L(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m \mid \exists \mathbf{s} \in \mathbb{Z}^n : \mathbf{y} = \mathbf{A}\mathbf{s} \bmod q\}$ and $L^\perp(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m \mid \mathbf{y}^T \mathbf{A} = \mathbf{0} \bmod q\}$.

The distance between a lattice L and a vector $\mathbf{v} \in \mathbb{R}^m$ is defined as $\text{dist}(\mathbf{v}, L) = \min\{\|\mathbf{v} - \mathbf{x}\| \mid \mathbf{x} \in L\}$. Furthermore, the i -th successive minimum $\lambda_i(L)$ of the lattice L is defined as the smallest radius r , such that there are i linearly independent vectors of norm at most r in the lattice. Let L be an m -dimensional lattice. Then the Gaussian heuristic is given as $\lambda_i(L) \approx \sqrt{\frac{m}{2\pi e}} \det(L)^{\frac{1}{m}}$ and the Hermite factor of a basis is given as $\delta_0^m = \frac{\|\mathbf{v}\|}{\det(L)^{\frac{1}{m}}}$, where \mathbf{v} is the shortest non-zero vector in the basis. The Hermite factor describes the quality of a basis, which, for example, may be the output of a basis reduction algorithm. δ_0 is called root-Hermite factor and $\log \delta_0$ the log-root-Hermite factor.

At last we define the fundamental parallelepiped as follows. Let \mathbf{X} be a set of n vectors \mathbf{x}_i . The fundamental parallelepiped of \mathbf{X} is defined as

$$P_{\frac{1}{2}}(\mathbf{X}) = \left\{ \sum_{i=0}^{n-1} \alpha_i \mathbf{x}_i \mid \alpha_i \in [-1/2, 1/2) \right\}$$

2.3 The LWE Problem and Solving Strategies

In the following we recall the definition of LWE.

Definition 1 (Learning with Errors Distribution)

Let n and $q > 0$ be integers, and $\alpha > 0$. We define by $\chi_{\mathbf{s}, \alpha}$ the LWE distribution which outputs $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{a} \leftarrow_{\S} \mathbb{Z}_q^n$ and $e \leftarrow_{\S} D_{\mathbb{Z}, \alpha q}$.

Definition 2 (Learning with Errors Problem) Let $n, m, q > 0$ be integers and $\alpha > 0$. Let the coefficients of \mathbf{s} be sampled according to $D_{\mathbb{Z}, \alpha q}$. Given m samples $(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ from $\chi_{\mathbf{s}, \alpha}$ for $i = 1, \dots, m$, the learning with errors problem is to find \mathbf{s} . Given m samples $(\mathbf{a}_i, c_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ for $i = 1, \dots, m$, the decisional learning with errors problem is to decide whether they are sampled by an oracle $\chi_{\mathbf{s}, \alpha}$ or whether they are sampled uniformly random in \mathbb{Z}_q .

In Regev's original definition of LWE, the attacker has access to arbitrary many LWE samples, which means that $\chi_{\mathbf{s}, \alpha}$ is seen as an oracle that outputs samples at will. If the maximum number of samples available is fixed, we can write them as a fixed set of $m > 0 \in \mathbb{Z}$ samples $\{(\mathbf{a}_1, c_1 = \mathbf{a}_1 \cdot \mathbf{s} + e_1 \bmod q), \dots, (\mathbf{a}_m, c_m = \mathbf{a}_m \cdot \mathbf{s} + e_m \bmod q)\}$, often written as matrix $(\mathbf{A}, \mathbf{c}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ with $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q$. We call \mathbf{A} *sample matrix*.

In the original definition, \mathbf{s} is sampled uniformly at random in \mathbb{Z}_q^n . At the loss of n samples, an LWE instance can be constructed where the secret \mathbf{s} is distributed as the error \mathbf{e} [7].

Two characterizations of LWE are considered in this work: the generic characterization by n, α, q , where the coefficients of secret and error are chosen according to the distribution $D_{\mathbb{Z}, \alpha q}$. The second characterization is LWE with small secret, i.e., the coefficients of the secret vector are chosen according to a distribution over a small set, e.g., $I = \{0, 1\}$; the error is again chosen with distribution $D_{\mathbb{Z}, \alpha q}$.

Learning with Errors Problem with Small Secret. In

the following, let $\{a, \dots, b\}$ be the set the coefficients of \mathbf{s} are sampled from for LWE instances with small secret. To solve LWE instances with small secret, some algorithms use *modulus switching*. Let $(\mathbf{a}, c = \mathbf{a} \cdot \mathbf{s} + e \bmod q)$ be a sample of an (n, α, q) -LWE instance. If the entries of \mathbf{s} are small enough, this sample can be transformed into a sample $(\tilde{\mathbf{a}}, \tilde{c})$ of an (n, α', p) -LWE instance, with $p < q$ and $\left\| \left(\left[\frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rfloor \right] \cdot \mathbf{s} \right) \right\| \approx \frac{p}{q} \|e\|$.

The transformed samples can be constructed such that $(\tilde{\mathbf{a}}, \tilde{c}) = \left(\left[\frac{p}{q} \cdot \mathbf{a} \right], \left[\frac{p}{q} \cdot c \right] \right) \in \mathbb{Z}_p^n \times \mathbb{Z}_p$ with

$$p \approx \sqrt{\frac{2\pi n}{12}} \cdot \frac{\sigma_{\mathbf{s}}}{\alpha} \quad (1)$$

and $\sigma_{\mathbf{s}}$ being the standard deviation of the elements of the secret vector \mathbf{s} [4, Lemma 2]. With the components of \mathbf{s} being uniformly distributed, the variance

of the elements of the secret vector \mathbf{s} is determined by $\sigma_{\mathbf{s}}^2 = \frac{(b-a+1)^2-1}{12}$. The result is an LWE instance with errors having standard deviation $\frac{\sqrt{2\alpha p}}{\sqrt{2\pi}} + \mathcal{O}(1)$ and $\alpha' = \sqrt{2}\alpha$. For some algorithms, such as the decoding attack or embedding approaches (cf. Section 4.2, 4.3, and 4.4, respectively), modulus switching should be combined with exhaustive search guessing g components of the secret at first. Then, the algorithm runs in dimension $n - g$. Therefore, all of these algorithms can be adapted to have at most the cost of exhaustive search and potentially have an optimal g somewhere in between zero and n .

The two main hardness assumptions leading to the basic strategies of solving LWE are the Short Integer Solutions (SIS) problem and the Bounded Distance Decoding (BDD) problem. We describe both of them in the following.

Short Integer Solutions Problem. The Short Integer Solutions (SIS) problem is defined as follows: Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ consisting of n vectors $\mathbf{a}_i \leftarrow_{\$} \mathbb{Z}_q^m$. Find a vector $\mathbf{v} \neq \mathbf{0} \in \mathbb{Z}^m$, such that $\|\mathbf{v}\| \leq \beta$ with $\beta < q \in \mathbb{Z}$ and $\mathbf{v}^T \mathbf{A} = \mathbf{0} \pmod{q}$.

Solving the SIS-problem with appropriate parameters solves Decision-LWE. Given m samples written as (\mathbf{A}, \mathbf{c}) , which either satisfy $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}$ or \mathbf{c} is chosen uniformly at random in \mathbb{Z}_q^m , the two cases can be distinguished by finding a short vector \mathbf{v} in the dual lattice $L^\perp(\mathbf{A})$. Then, $\mathbf{v} \cdot \mathbf{c}$ either results in $\mathbf{v} \cdot \mathbf{e}$, if $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}$, or $\mathbf{v} \cdot \mathbf{c}$ is uniformly random over \mathbb{Z}_q . In the first case, $\mathbf{v} \cdot \mathbf{c} = \mathbf{v} \cdot \mathbf{e}$ follows a Gaussian distribution over \mathbb{Z} , inherited from the distribution of \mathbf{e} , and is usually small. Therefore, as long as the Gaussian distribution can be distinguished from uniformly random, Decision-LWE can be solved as long as \mathbf{v} is short enough.

Bounded Distance Decoding Problem. The BDD problem is defined as follows. Given a lattice L , a target vector $\mathbf{c} \in \mathbb{Z}^m$, and $\text{dist}(\mathbf{c}, L) < \mu \lambda_1(L)$ with $\mu \leq \frac{1}{2}$. Find the lattice vector $\mathbf{x} \in L$ closest to \mathbf{c} .

An LWE instance $(\mathbf{A}, \mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q})$ can be seen as an instance of BDD. Let \mathbf{A} define the lattice $L(\mathbf{A})$. Then the point $\mathbf{w} = \mathbf{A}\mathbf{s}$ is contained in the lattice $L(\mathbf{A})$. Since \mathbf{e} follows the Gaussian distribution, over 99.7% of all encountered errors are within three standard deviations of the mean. For LWE parameters typically used in cryptographic applications, this is significantly smaller than $\lambda_1(L(\mathbf{A}))$. Therefore, \mathbf{w} is the closest lattice point to \mathbf{c} with very high probability. Hence, finding \mathbf{w} eliminates \mathbf{e} . If \mathbf{A} is invertible the secret \mathbf{s} can be calculated.

3 Description of basis reduction Algorithms

basis reduction is a very important building block of most of the algorithms to solve LWE considered in this paper. It is applied to a lattice L to find a basis $\{\mathbf{b}_0, \dots, \mathbf{b}_{n-1}\}$ of L , such that the basis vectors \mathbf{b}_i are short and nearly orthogonal to each other. Essentially, two different approaches to reduce a lattice basis are important in practice: the Lenstra-Lenstra-Lovász (LLL) basis reduction algorithm [14, 23, 27] and the Blockwise Korkine-Zolotarev (BKZ) algorithm with its improvements, called BKZ 2.0 [14, 17]. The runtime estimations of basis reduction used to solve LWE is independent of the number of given LWE samples. Hence, we do not describe the mentioned basis reduction algorithms but only summarize the runtime estimations used in the LWE-Estimator [4]. For a deeper contemplation, we refer to [4, 21, 24, 31].

Following the convention of Albrecht et al. [4], we assume that the first non-zero vector \mathbf{b}_0 of the basis of the reduced lattice is the shortest vector in the basis.

The Lenstra-Lenstra-Lovász Algorithm. Let L be a lattice with basis $\mathbf{B} = \{\mathbf{b}_0, \dots, \mathbf{b}_{n-1}\}$. Let $\mathbf{B}^* = \{\mathbf{b}_0^*, \dots, \mathbf{b}_{n-1}^*\}$ be the Gram-Schmidt basis with Gram-Schmidt coefficients $\mu_{i,j} = \frac{\mathbf{b}_i \cdot \mathbf{b}_j^*}{\|\mathbf{b}_j^*\|}$, $1 \leq j < i < n$. Let $\epsilon > 0$. Then the runtime of the LLL algorithm is determined by $\mathcal{O}(n^{5+\epsilon} \log^{2+\epsilon} B)$ with $B > \|\mathbf{b}_i\|$ for $0 \leq i \leq n-1$. Additionally, an improved variant, called L2, exists, whose runtime is estimated to be $\mathcal{O}(n^{5+\epsilon} \log B + n^{4+\epsilon} \log^2 B)$ [27]. Furthermore, a runtime $\mathcal{O}(n^3 \log^2 B)$ is estimated heuristically [14]. The first vector of the output basis is guaranteed to satisfy $\|\mathbf{b}_0\| \leq \left(\frac{4}{3} + \epsilon\right)^{\frac{n-1}{2}} \cdot \lambda_1(L)$ with $\epsilon > 0$.

The Blockwise Korkine-Zolotarev Algorithm. The BKZ algorithm employs an algorithm to solve several SVP instances of smaller dimension, which can be seen as an SVP oracle. The SVP oracle can be implemented by computing the Voronoi cells of the lattice, by sieving, or by enumeration. During BKZ several BKZ rounds are done. In each BKZ round an SVP oracle is called several times to receive a better basis after each round. The algorithm terminates when the quality of the basis remains unchanged after another BKZ round. The difference between BKZ and BKZ 2.0 are the usage of extreme pruning [17], early termination, limiting the enumeration radius to the Gaussian Heuristic, and local block pre-processing [14].

There exist several practical estimations of the runtime t_{BKZ} of BKZ in the literature. Some of these results are listed in the following. Lindner and Peikert's [24] estimation is given by $\log t_{\text{BKZ}}(\delta_0) = \frac{1.8}{\log \delta_0} - 78.9$ clock cycles, called LP model. This result should be used carefully, since applying this estimation implies the existence of a subexponential algorithm for solving LWE [4]. The estimation shown by Albrecht et al. [1] $\log t_{\text{BKZ}}(\delta_0) = \frac{0.009}{\log^2 \delta_0} - 4.1$, called delta-squared model, is non-linear in $\log \delta_0$ and it is claimed, that this is more suitable for current implementations. However, in the LWE-Estimator also a different approach is used. Given an n -dimensional lattice, the running time in clock cycles is estimated to be

$$t_{\text{BKZ}} = \rho \cdot n \cdot t_k, \quad (2)$$

where ρ is the number of BKZ rounds and t_k is the time needed to find short enough vectors in lattices of dimension k . Even though, ρ is exponentially upper bounded by $(nk)^n$ at best, in practice the results after $\rho = \frac{n^2}{k^2} \log n$ rounds yield a basis with $\|\mathbf{b}_0\| \leq 2\nu_k^{\frac{n-1}{2(k-1)} + \frac{3}{2}} \cdot \det(L)^{\frac{1}{n}}$, where $\nu_k \leq k$ is the maximum of root-Hermite factors in dimensions $\leq k$ [20]. However, recent results like progressive BKZ (running BKZ several times consecutively with increasing block sizes) show that even smaller values for ρ can be achieved. Consequently, the more conservative choice $\rho = 8$ is used in the LWE-Estimator. In the latest version of the LWE-Estimator the following runtime estimations to solve SVP of dimension k are used and compared¹:

$$t_{k,\text{enum}} = 0.27k \ln(k) - 1.02k + 16.10,$$

$$t_{k,\text{sieve}} = 0.29k + 16.40,$$

$$t_{k,\text{q-sieve}} = 0.27k + 16.40.$$

The estimation $t_{k,\text{enum}}$ are extrapolations of the runtime estimates presented by Chen and Nguyen [14]. The estimations $t_{k,\text{sieve}}$ and $t_{k,\text{q-sieve}}$ are presented in [11] and [22], respectively. The latter is a quantumly enhanced sieving algorithm.

Under the Gaussian heuristic and geometric series assumption, the following correspondence between the block size k and δ_0 can be given:

$$\lim_{n \rightarrow \infty} \delta_0 = \left(v_k^{\frac{-1}{k}} \right)^{\frac{1}{2(k-1)}} \approx \left(\frac{k}{2\pi e} (\pi k)^{\frac{1}{k}} \right)^{\frac{1}{2(k-1)}},$$

where v_k is the volume of the unit ball in dimension k . As examples show, this estimation may also be applied when n is finite [4]. As a function of k , the *lattice rule*

¹ $t_{k,\text{sieve}}$ is only applied for $\beta > 90$, which is true for all instances considered

of thumb approximates $\delta_0 = k^{\frac{1}{2k}}$, sometimes simplified to $\delta_0 = 2^{\frac{1}{k}}$. Albrecht et al. [4] show that the simplified lattice rule of thumb is a lower bound to the expected behavior on the interval [40, 250] of usual values for k . The simplified lattice rule of thumb is indeed closer to the expected behavior than the lattice rule of thumb, but it implies an subexponential algorithm for solving LWE. For later reference we write $\delta_0^{(1)} = \left(\frac{k}{2\pi e} (\pi k)^{\frac{1}{k}} \right)^{\frac{1}{2(k-1)}}$, $\delta_0^{(2)} = k^{\frac{1}{2k}}$, and $\delta_0^{(3)} = 2^{\frac{1}{k}}$.

4 Description of Algorithms to Solve the Learning with Errors Problem

In this section we describe the algorithms used to estimate the hardness of LWE and analyze them regarding their computational cost. If there exists a small secret variant of an algorithm, the corresponding section is divided into general and small secret variant.

Since the goal of this paper is to investigate how the number of samples m influences the hardness of LWE, we restrict our attention to attacks that are practical for restricted m . This excludes Arora and Ge's algorithm and BKW, which require at least sub-exponential m . Furthermore, we do not include purely combinatorial attacks like exhaustive search or meet-in-the-middle, since there runtime is not influenced by m .

4.1 Distinguishing Attack

The distinguishing attack solves decisional LWE via the SIS strategy using basis reduction. For this, the dual lattice $L^\perp(\mathbf{A}) = \{\mathbf{w} \in \mathbb{Z}^m \mid \mathbf{w}^T \mathbf{A} = \mathbf{0} \pmod{q}\}$ is considered. The dimension of the dual lattice $L^\perp(\mathbf{A})$ is m , the rank is m , and $\det(L^\perp(\mathbf{A})) = q^n$ (with high probability) [26]. Basis reduction is applied to find a short vector in $L^\perp(\mathbf{A})$. The result is used as short vector \mathbf{v} in the SIS-problem to distinguish the Gaussian from the uniform distribution. By doing so, the decisional LWE problem is solved. Since this attack is in a dual lattice, it is sometimes also called dual attack.

4.1.1 General Variant of the Distinguishing Attack

The success probability ϵ is the advantage of distinguishing $\mathbf{v} \cdot \mathbf{e}$ from uniformly random and can be approximated by standard estimates [4]

$$\epsilon = e^{-\pi(\|\mathbf{v}\|_\alpha)^2}. \quad (3)$$

| Model | Logarithmic runtime |
|---------------|-----------------------------------------------------------------------------------------------------|
| LP | $\frac{1.8m^2}{m \log\left(\frac{1}{\alpha} f(\epsilon)\right) - n \log(q)} - 78.9$ |
| delta-squared | $\frac{0.009m^4}{\left(m \log\left(\frac{1}{\alpha} f(\epsilon)\right) - n \log(q)\right)^2} + 4.1$ |

Table 1: Logarithmic runtime of the distinguishing attack for the LP and the delta-squared model (cf. Section 3)

| Relation δ_0 | Block size k in $t_{\text{BKZ}} = \rho \cdot n \cdot t_k$, cf. Eq. (2) |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\delta_0^{(1)}$ | $\frac{\log\left(\frac{k}{2\pi\epsilon} (\pi k)^{\frac{1}{k}}\right)}{2(k-1)} = \frac{\log\left(\frac{f(\epsilon)}{\alpha}\right)}{m} - \frac{n \log q}{m^2}$ |
| $\delta_0^{(2)}$ | $\frac{k}{\log k} = \frac{1}{2} \frac{m^2}{m \log\left(\frac{1}{\alpha} f(\epsilon)\right) - n \log(q)}$ |
| $\delta_0^{(3)}$ | $k = \frac{m}{\log\left(\frac{1}{\alpha} f(\epsilon)\right) - \frac{n}{m} \log(q)}$ |

Table 2: Block size k depending on δ_0 required to achieve a success probability of ϵ for the distinguishing attack for different models for the relation of k and δ_0 (cf. Section 3)

In order to achieve a fixed success probability ϵ , a vector \mathbf{v} of length $\|\mathbf{v}\| = \frac{1}{\alpha} \sqrt{\ln\left(\frac{1}{\epsilon}\right)}/\pi$ is needed. Let $f(\epsilon) = \sqrt{\ln\left(\frac{1}{\epsilon}\right)}/\pi$. The logarithm of δ_0 required to achieve a success probability of ϵ to distinguish $\mathbf{v} \cdot \mathbf{e}$ from uniformly random is given as

$$\log \delta_0 = \frac{1}{m} \log\left(\frac{1}{\alpha} f(\epsilon)\right) - \frac{n}{m^2} \log(q), \quad (4)$$

where m is the given number of LWE samples. To estimate the runtime of the distinguishing attack, it is sufficient to determine δ_0 , since the attack solely depends on basis reduction. Table 1 gives the runtime estimations of the distinguishing attack in the LP and the delta-squared model described in Section 3. Table 2 gives the block size k of BKZ derived in Section 3 following the second approach to estimate the runtime of the distinguishing attack.

On the one hand, the runtime of BKZ decreases exponentially with the length of \mathbf{v} . On the other hand, using a longer vector reduces the success probability. To achieve an overall success probability close to 1, the algorithm has to be run multiple times. The number of repetitions is determined to be $\frac{1}{\epsilon^2}$ via the Chernoff bound [15]. Let $T(\epsilon, m)$ be the runtime of a single execution of the algorithm. Then, the best overall runtime is the minimum of $\frac{1}{\epsilon^2} T(\epsilon, m)$ over different choices of ϵ . This requires randomization of the attack to achieve independent runs. We assume that an attacker can achieve this without using additional samples, which is conservative from an cryptographic point of view.

| Model | Logarithmic runtime |
|---------------|-------------------------------------------------------------------------------------------------------------|
| LP | $\frac{1.8m^2}{m \log\left(\frac{1}{\sqrt{2}\alpha} f(\epsilon)\right) - n \log(p)} - 78.9$ |
| delta-squared | $\frac{0.009m^4}{\left(m \log\left(\frac{1}{\sqrt{2}\alpha} f(\epsilon)\right) - n \log(p)\right)^2} + 4.1$ |

Table 3: Logarithmic runtime of the distinguishing attack with small secret in the LP and the delta-squared model (cf. Section 3)

| Relation δ_0 | Block size k in $t_{\text{BKZ}} = \rho \cdot n \cdot t_k$, cf. Eq. (2) |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\delta_0^{(1)}$ | $\frac{\log\left(\frac{k}{2\pi\epsilon} (\pi k)^{\frac{1}{k}}\right)}{2(k-1)} = \frac{\log\left(\frac{1}{\sqrt{2}\alpha} f(\epsilon)\right)}{m} - \frac{n \log(p)}{m^2}$ |
| $\delta_0^{(2)}$ | $\frac{k}{\log k} = \frac{1}{2} \frac{m^2}{m \log\left(\frac{1}{\sqrt{2}\alpha} f(\epsilon)\right) - n \log(p)}$ |
| $\delta_0^{(3)}$ | $k = \frac{m}{m \log\left(\frac{1}{\sqrt{2}\alpha} f(\epsilon)\right) - n \log(p)}$ |

Table 4: Block size k depending on δ_0 required to achieve a success probability of ϵ for the distinguishing attack with small secret for different models for the relation of k and δ_0 (cf. Section 3)

4.1.2 Small Secret Variant of the Distinguishing Attack

The distinguishing attack for small secrets works similar to the general case, but it exploits the smallness of the secret \mathbf{s} by applying modulus switching at first. To solve a small secret LWE instance with the distinguishing attack, the strategy described in Section 2.3 can be applied: First, modulus switching is used and afterwards the algorithm is combined with exhaustive search.

Using the same reasoning as in the standard case, the required $\log \delta_0$ for an $n, \sqrt{2}\alpha, p$ -LWE instance is given by

$$\log \delta_0 = \frac{1}{m} \log\left(\frac{1}{\sqrt{2}\alpha} f(\epsilon)\right) - \frac{n}{m^2} \log p, \quad (5)$$

where p can be estimated by Equation (1). The rest of the algorithm remains the same as in the standard case. Table 3 gives the run times estimations of in the LP and the delta-squared model described in Section 3. Table 4 gives the block size k of BKZ derived in Section 3 following the second approach to estimate run times of the distinguishing attack with small secret. Combining this algorithm with exhaustive search as described in Section 2.3 may improve the runtime.

4.2 Decoding Approach

The decoding approach solves LWE via the BDD strategy described in Section 2. The procedure considers the lattice $L = L(\mathbf{A})$ defined by the sample matrix \mathbf{A} and consists of two steps: the reduction step and the decoding step. In the reduction step, basis reduction is employed on L . In the decoding phase the resulting basis is used to find a close lattice vector $\mathbf{w} = \mathbf{A}\mathbf{s}$ and thereby eliminate the error vector \mathbf{e} .

In the following let the *target success probability* be the overall success probability of the attack, chosen by the attacker (usually close to 1). In contrast, the success probability refers to the success probability of a single run of the algorithm. The target success probability is achieved by running the algorithm potentially multiple times with a certain success probability for each single run.

4.2.1 General Variant of Decoding Approach

To solve BDD, and therefore LWE, the most basic algorithm is Babai's Nearest Plane algorithm [9]. Given a BDD instance $(\mathbf{A}, \mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q)$ from m samples, the solving algorithm consists of two steps. First, basis reduction on the lattice $L = L(\mathbf{A})$ is used, which results in a new basis $\mathbf{B} = (\mathbf{b}_0, \dots, \mathbf{b}_{n-1})$ for L with root-Hermite factor δ_0 . The decoding steps is a recursive algorithm that gets as input the partial basis $\mathbf{B}' = (\mathbf{b}_0, \dots, \mathbf{b}_{n-i})$ (the complete basis in the first call) and a target vector \mathbf{v} (\mathbf{c} in the first call). In every step, it searches for the coefficient α_{n-i} such that $\mathbf{v}' = \mathbf{v} - \alpha_{n-i}\mathbf{b}_{n-i}$ is as close as possible to the subspace spanned by $(\mathbf{b}_0, \dots, \mathbf{b}_{n-i-1})$. The recursive call is then with the new sub-basis $(\mathbf{b}_0, \dots, \mathbf{b}_{n-i-1})$ and \mathbf{v}' as target vector.

The result of the algorithm is the lattice point $\mathbf{w} \in L$, such that $\mathbf{c} \in \mathbf{w} + P_{\frac{1}{2}}(\mathbf{B}^*)$. Therefore, the algorithm is able to recover \mathbf{s} correctly from $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q$ if and only if \mathbf{e} lies in the fundamental parallelepiped $P_{\frac{1}{2}}(\mathbf{B}^*)$. The success probability of the Nearest Plane algorithm is the probability of \mathbf{e} falling into $P_{\frac{1}{2}}(\mathbf{B}^*)$:

$$\begin{aligned} \Pr \left[\mathbf{e} \in P_{\frac{1}{2}}(\mathbf{B}^*) \right] &= \prod_{i=0}^{m-1} \Pr \left[|\mathbf{e} \cdot \mathbf{b}_i^*| < \frac{\mathbf{b}_i^* \cdot \mathbf{b}_i^*}{2} \right] \\ &= \prod_{i=0}^{m-1} \operatorname{erf} \left(\frac{\|\mathbf{b}_i^*\| \sqrt{\pi}}{2\alpha q} \right). \end{aligned} \quad (6)$$

Hence, an attacker can adjust his overall runtime according to the trade-off between the quality of the basis reduction and the success probability.

Lindner and Peikert [24] present a modification of the Nearest Plane algorithm named Nearest Planes. They introduce additional parameters $d_i \geq 1$ to the decoding step, which describes how many nearest planes the algorithm takes into account on the i -th level of recursion.

The success probability of the Nearest Planes algorithm is the probability of \mathbf{e} falling into the parallelepiped $P_{\frac{1}{2}}(\mathbf{B}^* \operatorname{diag}(\mathbf{d}))$, given as follows:

$$\Pr \left[\mathbf{e} \in P_{\frac{1}{2}}(\mathbf{B}^* \operatorname{diag}(\mathbf{d})) \right] = \prod_{i=0}^{m-1} \operatorname{erf} \left(\frac{d_i \|\mathbf{b}_i^*\| \sqrt{\pi}}{2\alpha q} \right). \quad (7)$$

To choose values d_i , Lindner and Peikert suggest to maximize $\min(d_i \|\mathbf{b}_i^*\|)$ while minimizing the overall runtime. As long as the values d_i are powers of 2, this can be shown to be optimal [4]. For a fixed success probability, the optimal values d_i can be found iteratively. In each iteration, the value d_i , for which $d_i \|\mathbf{b}_i^*\|$ is currently minimal, is usually increased by one. Then, the success probability given by Equation (7) is calculated again. If the result is at least as large as the chosen success probability, the iteration stops [24]. An attacker can choose the parameters δ_0 and d_i , which determine the success probability ϵ of the algorithm. Presumably, an attacker tries to minimize the overall runtime

$$T = \frac{T_{\text{BKZ}} + T_{\text{NP}}}{\epsilon}, \quad (8)$$

where T_{BKZ} is the runtime of the basis reduction with chosen target quality δ_0 , T_{NP} is the runtime of the decoding step with chosen d_i and ϵ is the success probability achieved by δ_0 and d_i . To estimate the overall runtime it is reasonable to assume that the time of the basis reduction and the decoding step are balanced. To give a more precise estimation, one bit has to be subtracted from the number of operations, since the estimation is up to a factor of 2 worse than the optimal runtime.

The runtime of the basis reduction is determined by δ_0 as described in Section 3. The values d_i cannot be expressed by a formula and therefore, there is also no closed formula for δ_0 . As a consequence, the runtime of the basis reduction step cannot be explicitly given here. They are found by iteratively varying values for δ_0 until the running times of the two steps are balanced as described above.

The runtime of the decoding step for Lindner and Peikert's Nearest Planes algorithm, is determined by the number of points $\prod_{i=0}^{m-1} d_i$ that have to be exhausted and the time t_{node} it takes to process one point:

$$T_{\text{NP}} = t_{\text{node}} \prod_{i=0}^{m-1} d_i. \quad (9)$$

Since no closed formula is known to calculate the values d_i , they are computed by step-wise increasing like described above until the success probability calculated by Equation (7) reaches the fixed success probability. In the LWE-Estimator $t_{\text{node}} \approx 10^{15.1}$ clock cycles is used. Hence, both, the runtime T_{BKZ} and T_{NP} , depend on δ_0 and the fixed success probability.

Since this contemplation only considers a fixed success probability, the best trade-off between success probability and the running time of a single execution described above must be found by repeating the process above with varying values of the fixed success probability.

4.2.2 Small Secret Variant of Decoding Approach

The decoding approach for small secrets works the same as in the general case, but it exploits the smallness of the secret \mathbf{s} by applying modulus switching at first and combining this algorithm with exhaustive search afterwards as described in Section 2.3.

4.3 Standard Embedding

The standard embedding attack solves LWE via reduction to uSVP. The reduction is done by creating an $m + 1$ -dimensional lattice that contains the error vector \mathbf{e} . Since \mathbf{e} is very short for typical instantiations of LWE, this results in a uSVP instance. The typical way to solve uSVP is to apply basis reduction.

Let $(\mathbf{A}, \mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q)$ and $t = \text{dist}(\mathbf{c}, L(\mathbf{A})) = \|\mathbf{c} - \mathbf{x}\|$ where $\mathbf{x} \in L(\mathbf{A})$, such that $\|\mathbf{c} - \mathbf{x}\|$ is minimized. Then the lattice $L(\mathbf{A})$ can be embedded in the lattice $L(\mathbf{A}')$, with

$$\mathbf{A}' = \begin{pmatrix} \mathbf{A} & \mathbf{c} \\ \mathbf{0} & t \end{pmatrix}. \quad (10)$$

If $t < \frac{\lambda_1(L(\mathbf{A}))}{2\gamma}$, the higher-dimensional lattice $L(\mathbf{A}')$ has a unique shortest vector $\mathbf{c}' = (-\mathbf{e}, t) \in \mathbb{Z}_q^{m+1}$ with length $\|\mathbf{c}'\| = \sqrt{m\alpha^2q^2/(2\pi) + |t|^2}$ [25, 32]. Therefore, \mathbf{e} can be extracted from \mathbf{c}' , $\mathbf{A}\mathbf{s}$ is known, and \mathbf{s} can be solved for.

To determine the success probability and the runtime, we distinguish between two case: $t = \|\mathbf{e}\|$ and $t < \|\mathbf{e}\|$. The case $t = \|\mathbf{e}\|$ is mainly of theoretical interest. Practical attacks and the LWE-Estimator use $t = 1$ instead, so we focus on this case in the following.

| Model | Logarithmic runtime |
|---------------|-----------------------------------------------------------------------------------------------------|
| LP | $\frac{1.8m}{\log(q^{1-\frac{n}{m}}) - \log(\sqrt{\epsilon}\tau\alpha q)} - 78.9$ |
| delta-squared | $\frac{0.009m^2}{\left(\log(q^{1-\frac{n}{m}}) - \log(\sqrt{\epsilon}\tau\alpha q)\right)^2} + 4.1$ |

Table 5: Logarithmic runtime of the standard embedding attack in the LP and the delta-squared model (cf. Section 3)

| Relation δ_0 | Block size k in $t_{\text{BKZ}} = \rho \cdot n \cdot t_k$, cf. Eq. (2) |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\delta_0^{(1)}$ | $\left(\frac{k}{2\pi e} (\pi k)^{\frac{1}{k}}\right)^{\frac{1}{2(k-1)}} = \left(\frac{q^{1-\frac{n}{m}} \sqrt{\frac{1}{e}}}{\tau\alpha q}\right)^{\frac{1}{m}}$ |
| $\delta_0^{(2)}$ | $\frac{k}{\log k} = \frac{1}{2} \frac{m}{\log(q^{1-\frac{n}{m}}) - \log(\sqrt{\epsilon}\tau\alpha q)}$ |
| $\delta_0^{(3)}$ | $k = \frac{m}{\log(q^{1-\frac{n}{m}}) - \log(\sqrt{\epsilon}\tau\alpha q)}$ |

Table 6: Block size k depending on δ_0 required such that the standard embedding attack succeeds for different models for the relation of k and δ_0 (cf. Section 3)

Based on Albrecht et al. [2], Göpfert shows [19, Section 3.1.3] that the standard embedding attack succeeds with non-negligible probability if

$$\delta_0 \leq \left(\frac{q^{1-\frac{n}{m}} \sqrt{\frac{1}{e}}}{\tau\alpha q}\right)^{\frac{1}{m}}, \quad (11)$$

where m is the number of LWE samples. The value τ is experimentally determined to be $\tau \leq 0.4$ for a success probability of $\epsilon = 0.1$ [2].

In Table 5, we put all together and state the runtime for the cases from Section 3 of the standard embedding attack in the LP and the delta-squared model. Table 6 gives the block size k of BKZ derived in Section 3 following the second approach to estimate run times of the standard embedding attack.

As discussed above, the success probability ϵ of a single run depends on τ and thus does not necessarily yield the desired target success probability ϵ_{target} . If the success probability is lower than the target success probability, the algorithm has to be repeated ρ times to achieve

$$\epsilon_{\text{target}} \leq 1 - (1 - \epsilon)^\rho. \quad (12)$$

Consequently, it has to be considered that ρ executions of this algorithm have to be done, i.e., the runtime has to be multiplied by ρ . As before we assume that the samples may be reused in each run.

| Model | Logarithmic runtime |
|---------------|----------------------------------------------------------------------------------------------------------|
| LP | $\frac{1.8m}{\log\left(p^{1-\frac{n}{m}}\right) - \log(\sqrt{2e\tau\alpha p})} - 78.9$ |
| delta-squared | $\frac{0.009m^2}{\left(\log\left(p^{1-\frac{n}{m}}\right) - \log(\sqrt{2e\tau\alpha p})\right)^2} + 4.1$ |

Table 7: Logarithmic runtime of the standard embedding attack with small secret in the LP and the delta-squared model (cf. Section 3)

4.3.1 Small Secret Variant of Standard Embedding

To solve a small secret LWE instance based on embedding, the strategy described in Section 2.3 can be applied: First, modulus switching is used and afterwards the algorithm is combined with exhaustive search. The standard embedding attack on LWE with small secret using modulus switching works the same as standard embedding in the non-small secret case, except that it operates on instances characterized by n , $\sqrt{2}\alpha$, and p instead of n , α , and q with $p < q$. It is combined with guessing parts of the secret, which allows for larger δ_0 and therefore for an easier basis reduction. To be more precise, the requirement for δ_0 from Equation (11) changes as follows:

$$\delta_0 \leq \left(\frac{p^{1-\frac{n}{m}} \sqrt{\frac{1}{e}}}{\tau \sqrt{2}\alpha p} \right)^{\frac{1}{m}}, \quad (13)$$

where p can be estimated by Equation (1). As stated in the description of the standard case, the overall runtime of the algorithm is determined depending on δ_0 .

In Table 7, we state the runtime of the standard embedding attack in the LP and the delta-squared model. Table 8 gives the block size k of BKZ derived in Section 3 following the second approach to estimate runtime of the standard embedding attack with small secret. The success probability remains the same.

4.4 Dual Embedding

Dual embedding is very similar to standard embedding shown in Section 4.3. However, since the embedding is into a different lattice, the dual embedding algorithm runs in dimension $n + m + 1$ instead of $m + 1$, while the number of required samples remains m . Therefore, it is more suitable for instances with a restricted number

| Relation δ_0 | Block size k in $t_{\text{BKZ}} = \rho \cdot n \cdot t_k$, cf. Eq. (2) |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\delta_0^{(1)}$ | $\left(\frac{k}{2\pi e} (\pi k)^{\frac{1}{k}} \right)^{\frac{1}{2(k-1)}} = \left(\frac{p^{1-\frac{n}{m}} \sqrt{\frac{1}{e}}}{\tau \sqrt{2}\alpha p} \right)^{\frac{1}{m}}$ |
| $\delta_0^{(2)}$ | $\frac{k}{\log k} = \frac{1}{2} \frac{m}{\log\left(p^{1-\frac{n}{m}}\right) - \log(\sqrt{2e\tau\alpha p})}$ |
| $\delta_0^{(3)}$ | $k = \frac{m}{\log\left(p^{1-\frac{n}{m}}\right) - \log(\sqrt{2e\tau\alpha p})}$ |

Table 8: Block size k depending on δ_0 required such that the standard embedding attack with small secret succeeds for different models for the relation of k and δ_0 (cf. Section 3)

of LWE samples [32]. In case the optimal number of samples is given (as assumed in the LWE-Estimator by Albrecht et al.) this attack is as efficient as the standard embedding attack. Hence, it was not included in the LWE-Estimator so far.

4.4.1 General Variant of Dual Embedding

For an LWE instance $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}$, let the matrix $\mathbf{A}_o \in \mathbb{Z}_q^{m \times (n+m+1)}$ be defined as

$$\mathbf{A}_o = \begin{pmatrix} \mathbf{A} & \mathbf{I}_m & \mathbf{c} \end{pmatrix},$$

with $\mathbf{I}_m \in \mathbb{Z}^{m \times m}$ being the identity matrix. Define $L^\perp(\mathbf{A}_o) = \{\mathbf{v} \in \mathbb{Z}^{n+m+1} \mid \mathbf{A}_o \mathbf{v} = \mathbf{0} \pmod{q}\}$ to be the lattice in which uSVP is solved. Considering $\mathbf{v} = (\mathbf{s}, \mathbf{e}, -1)^T$ leads to $\mathbf{A}_o \mathbf{v} = \mathbf{A}\mathbf{s} + \mathbf{e} - \mathbf{c} = \mathbf{0} \pmod{q}$ and therefore $\mathbf{v} \in L^\perp(\mathbf{A}_o)$. The length of \mathbf{v} is small and can be estimated to be $\|\mathbf{v}\| \approx \sqrt{(n+m)\alpha^2 q^2 / \sqrt{2\pi}}$ [32].

Since this attack is similar to standard embedding, the estimations of the success probability and the running time is the same except for adjustments with respect to the dimension and determinant. Hence, the dual embedding attack is successful if the root-Hermite delta fulfills

$$\delta_0 = \left(\frac{q^{\frac{m}{m+n}} \sqrt{\frac{1}{e}}}{\tau \alpha q} \right)^{\frac{1}{n+m}}, \quad (14)$$

while the number of LWE samples is m .

In Table 9, we state the runtime of the dual embedding attack in the LP and the delta-squared model. Table 10 gives the block size k of BKZ derived in Section 3 following the second approach to estimate runtime of the dual embedding attack.

Since this algorithm is not mentioned by Albrecht et al. [4], we explain the analysis for an unlimited number of samples in the following. The case, where the number of

| Model | Logarithmic runtime |
|---------------|-------------------------------------------------------------------------------------------------------------|
| LP | $\frac{1.8(n+m)}{\log\left(q^{\frac{m}{m+n}}\right) - \log(\sqrt{e}\tau\alpha q)} - 78.9$ |
| delta-squared | $\frac{0.009(n+m)^2}{\left(\log\left(q^{\frac{m}{m+n}}\right) - \log(\sqrt{e}\tau\alpha q)\right)^2} + 4.1$ |

Table 9: Logarithmic runtime of the dual embedding attack in the LP and the delta-squared model (cf. Section 3)

| Relation δ_0 | Block size k in $t_{\text{BKZ}} = \rho \cdot n \cdot t_k$, cf. Eq. (2) |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\delta_0^{(1)}$ | $\left(\frac{k}{2\pi e}(\pi k)^{\frac{1}{k}}\right)^{\frac{1}{2(k-1)}} = \left(\frac{q^{\frac{m}{m+n}}\sqrt{\frac{1}{e}}}{\tau\alpha q}\right)^{\frac{1}{n+m}}$ |
| $\delta_0^{(2)}$ | $\frac{k}{\log k} = \frac{1}{2} \frac{n+m}{\log\left(q^{\frac{m}{m+n}}\right) - \log(\sqrt{e}\tau\alpha q)}$ |
| $\delta_0^{(3)}$ | $k = \frac{n+m}{\log\left(q^{\frac{m}{m+n}}\right) - \log(\sqrt{e}\tau\alpha q)}$ |

Table 10: Block size k depending on δ_0 required such that the dual embedding attack succeeds for different models for the relation of k and δ_0 (cf. Section 3)

samples is not limited and thus, the optimal number of samples can be used, is a special case of the discussion above. To be more precise, to find the optimal number of samples m_{optimal} , the parameter m with maximal δ_0 (according to Equation (14)) has to be found. This yields the lowest runtime using dual-embedding. The success probability is determined similar to the standard embedding, see Section 4.3.

4.4.2 Small Secret Variant of Dual Embedding

There are two small secret variant of the dual embedding attack: One is similar to the small secret variant of the standard embedding, the other is better known as the embedding attack by Bai and Galbraith. Both are described in the following.

Small Secret Variant of Dual Embedding with Modulus Switching. As before, the strategy described in Section 2.3 can be applied: First, modulus switching is used and afterwards the algorithm is combined with exhaustive search. This variant works the same as dual embedding in the non-small secret case, except that it operates on instances characterized by n , $\sqrt{2}\alpha$, and p instead of n , α , and q with $p < q$. This allows for larger δ_0 and therefore for an easier basis reduction. Hence,

the following inequality has to be fulfilled by δ_0 :

$$\delta_0 = \left(\frac{p^{\frac{m}{m+n}}\sqrt{\frac{1}{e}}}{\tau\sqrt{2}\alpha p}\right)^{\frac{1}{n+m}}, \quad (15)$$

where p can be estimated by Equation (1).

In Table 11, we state the runtime of the dual embedding attack with small secret in the LP and the delta-squared model. Table 12 gives the block size k of BKZ derived in Section 3 following the second approach to estimate runtime of the dual embedding attack with small secret. The success probability remains the same.

| Model | Logarithmic runtime |
|---------------|--------------------------------------------------------------------------------------------------------------|
| LP | $\frac{1.8(n+m)}{\log\left(p^{\frac{m}{m+n}}\right) - \log(\sqrt{2e}\tau\alpha p)} - 78.9$ |
| delta-squared | $\frac{0.009(n+m)^2}{\left(\log\left(p^{\frac{m}{m+n}}\right) - \log(\sqrt{2e}\tau\alpha p)\right)^2} + 4.1$ |

Table 11: Logarithmic runtime of the dual embedding attack with small secret using modulus switching in the LP and the delta-squared model (cf. Section 3)

| Relation δ_0 | Block size k in $t_{\text{BKZ}} = \rho \cdot n \cdot t_k$, cf. Eq. (2) |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\delta_0^{(1)}$ | $\left(\frac{k}{2\pi e}(\pi k)^{\frac{1}{k}}\right)^{\frac{1}{2(k-1)}} = \left(\frac{p^{\frac{m}{m+n}}\sqrt{\frac{1}{e}}}{\tau\sqrt{2}\alpha p}\right)^{\frac{1}{n+m}}$ |
| $\delta_0^{(2)}$ | $\frac{k}{\log k} = \frac{1}{2} \frac{n+m}{\log\left(p^{\frac{m}{m+n}}\right) - \log(\sqrt{2e}\tau\alpha p)}$ |
| $\delta_0^{(3)}$ | $k = \frac{n+m}{\log\left(p^{\frac{m}{m+n}}\right) - \log(\sqrt{2e}\tau\alpha p)}$ |

Table 12: Block size k depending on δ_0 required such that the dual embedding attack with small secret using modulus switching succeeds for different models for the relation of k and δ_0 (cf. Section 3)

Bai and Galbraith's Embedding. The embedding attack by Bai and Galbraith [10] solves LWE with a small secret vector \mathbf{s} , with each entry in $[a, b]$, by embedding. Similar to the dual embedding, Bai and Galbraith's solves uSVP in the lattice $L^\perp(\mathbf{A}_o) = \{\mathbf{v} \in \mathbb{Z}^{n+m+1} \mid \mathbf{A}_o \mathbf{v} = \mathbf{0} \pmod{q}\}$ for the matrix $\mathbf{A}_o \in \mathbb{Z}_q^{m \times (n+m+1)}$ defined as $\mathbf{A}_o = \begin{pmatrix} \mathbf{A} & \mathbf{I}_m & \mathbf{c} \end{pmatrix}$ in order to recover the short vector $\mathbf{v} = (\mathbf{s}, \mathbf{e}, -1)^T$. Since $\|\mathbf{s}\| \ll \|\mathbf{e}\|$, the uSVP algorithm has to find an unbalanced solution.

To tackle this, the lattice should be scaled such that it is more balanced, i.e., the first n rows of the lattice

| Model | Logarithmic runtime |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| LP | $\frac{1.8m'^2}{m' \log\left(\frac{q}{2\sigma\tau\sqrt{\pi e}}\right) + n \log\left(\frac{\xi\sigma}{q}\right)} - 78.9$ |
| delta-squared | $\frac{0.009m'^4}{\left(m' \log\left(\frac{q}{2\sigma\tau\sqrt{\pi e}}\right) + n \log\left(\frac{\xi\sigma}{q}\right)\right)^2} + 4.1$ |

Table 13: Logarithmic runtime of Bai and Galbraith’s embedding attack in the LP and the delta-squared model (cf. Section 3)

| Rel. δ_0 | Block size k in $t_{\text{BKZ}} = \rho \cdot n \cdot t_k$, cf. Eq. (2) |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\delta_0^{(1)}$ | $\left(\frac{k}{2\pi e}(\pi k)^{\frac{1}{k}}\right)^{\frac{1}{2(k-1)}} = \frac{m' \log\left(\frac{q}{2\sigma\tau\sqrt{\pi e}}\right) + n \log\left(\frac{\xi\sigma}{q}\right)}{m'^2}$ |
| $\delta_0^{(2)}$ | $\frac{k}{\log k} = \frac{1}{2} \frac{m'^2}{m' \log\left(\frac{q}{2\sigma\tau\sqrt{\pi e}}\right) + n \log\left(\frac{\xi\sigma}{q}\right)}$ |
| $\delta_0^{(3)}$ | $k = \frac{m'^2}{m' \log\left(\frac{q}{2\sigma\tau\sqrt{\pi e}}\right) + n \log\left(\frac{\xi\sigma}{q}\right)}$ |

Table 14: Block size k depending on δ_0 determined in the Embedding attack by Bai and Galbraith for different models for the relation of k and δ_0 (cf. Section 3)

basis are multiplied with a factor depending on σ [10]. Hence, the determinant of the lattice is increased by a factor of $\left(\frac{2}{b-a}\sigma\right)^n$ without significantly increasing the norm the error vector. This increases the δ_0 needed to successfully execute the attack. The required δ_0 can be determined similarly as done in the standard embedding in Section 4.3:

$$\log \delta_0 = \frac{m' \log\left(\frac{q}{2\sigma\tau\sqrt{\pi e}}\right) + n \log\left(\frac{\xi\sigma}{q}\right)}{m'^2}, \quad (16)$$

where $m' = n + m$, $\xi = 2/(b - a)$, and m LWE samples are used.

In Table 13, we state the runtime of the Bai-Galbraith embedding attack in the LP and the delta-squared model. Table 14 gives the block size k of BKZ derived in Section 3 following the second approach to estimate runtime of the Bai-Galbraith embedding attack with small secret.

The success probability is determined similar to the standard embedding, see Section 4.3, Equation (12).

Similar to the other algorithms for LWE with small secret, the runtime of Bai and Galbraith’s attack can be combined with exhaustive search guessing parts of the secret. However, in contrast to the other algorithms using basis reduction, Bai and Galbraith state that applying modulus switching to their algorithm does not improve the result. The reason for this is, that modulus switching reduces q by a larger factor than it reduces the size of the error.

5 Implementation

In this section, we describe our implementation of the results presented in Section 4 as an extension of the LWE-Estimator introduced by Albrecht et al. [3, 4]. Furthermore, we compare results of our implementation focusing on the behavior when limiting the number of available LWE samples.

5.1 Description of our Implementation

Our extension is also written in sage and it is already merged with the original LWE-Estimator (from commit-id eb45a74 on) in March 2017. In the following we used the version of the LWE-Estimator from June 2017 (commit-id: e0638ac) for our experiments.

Except for Arora and Ge’s algorithm based on Gröbner bases, we adapt each algorithm the LWE-Estimator implements to take a fixed number of samples into account if a number of samples is given by the user. If not, each of the implemented algorithms assumes unlimited number of samples (and hence assumes the optimal number of samples is available). Our implementation also extends the algorithms coded-BKW, decisional-BKW, search-BKW, and meet-in-the-middle attacks (for a description of these algorithms see [4]) although we omitted the theoretical description of these algorithms in Section 4.

Following the notation of Albrecht et al. [4], we assign an abbreviation to each algorithm to refer:

| | |
|-------------|---------------------------------------------------------------|
| dual | distinguishing attack, Sec. 4.1 |
| dec | decoding attack, Sec. 4.2 |
| usvp-primal | standard embedding, Sec. 4.3 |
| usvp-dual | dual embedding, Sec. 4.4 |
| usvp-baigal | Bai-Galbraith embedding, Sec. 4.4.2 |
| usvp | minimum of usvp-primal, usvp-dual, and usvp-baigal |
| mitm | exhaustive search ² |
| bkw | coded-BKW ³ |
| arora-gb | Arora and Ge’s algorithm based on Gröbner bases. ⁴ |

² Exhaustive search and meet-in-the-middle attacks are not discussed in Section 4 but they are adapted and implemented in our software.

³ coded-BKW, search-BKW, and decision-BKW are not discussed in Section 4 but they are adapted and implemented in the software.

⁴ Arora and Ge’s algorithm based on Gröbner bases is not adapted to a fixed number of samples in the implementation.

The shorthand symbol `bkw` solely refers to coded-BKW and its small secret variant. Decision-BKW and Search-BKW are not assigned an abbreviation and are not used by the main method `estimate_lwe`, because coded-BKW is the latest and most efficient BKW algorithm. Nevertheless, the other two BKW algorithms can be called separately via the function `bkw`, which is a convenience method for the functions `bkw_search` and `bkw_decision`, and its corresponding small secret variant `bkw_small_secret`.

In the LWE-Estimator the three different embedding approaches `usvp-primal`, `usvp-dual`, and `usvp-baigal` (in case of LWE with small secret is called) are summarized as the attack `usvp` and the minimum of the three embedding algorithms is returned. In our experiments we show the different impacts of those algorithms and hence we display the results of the three embedding approaches separately.

Let the LWE instance be defined by n , $\alpha = \frac{1}{\sqrt{2\pi n \log^2 n}}$, and $q \approx n^2$ as proposed by Regev [30]. In the following, let $n = 128$ and the number of samples be given by $m = 256$. If instead of α only the Gaussian width parameter (`sigma_is_stddev=False`) or the standard deviation (`sigma_is_stddev=True`) is known, α can be calculated by `alphaf(sigma, q, sigma_is_stddev)`.

The main function to call the LWE-Estimator is called `estimate_lwe`. Listing 1 shows how to call the LWE-Estimator on the given LWE instance (with Gaussian distributed error and secret) including the following attacks: distinguishing attack, decoding, and embedding attacks. The first two lines of Listing 1 define the parameters n, α, q , and the number of samples m . In the third line the program is called via `estimate_lwe`. For each algorithm a value `rop` is returned that gives the hardness of the LWE instance with respect to the corresponding attack.

```
sage: n,alpha,q = Param.Regev(128)
sage: m = 256
sage: costs=estimate_lwe(n,alpha,q,m=m,skip="arora-gb,bkw,mitm")
usvp: rop: ≈2^48.9, m: 226, red: ≈2^48.9, δ_0: 1.009971, β: 86, d: 355, repeat: 44
dec: rop: ≈2^56.8, m: 235, red: ≈2^56.8, δ_0: 1.009308, β: 99, d: 363, babai: ≈2^42.2, babai_op: ≈2^57.3, repeat: 146, ε: 0.031250
dual: rop: ≈2^74.7, m: 376, red: ≈2^74.7, δ_0: 1.008810, β: 111, d: 376, |v|: 736.521, repeat: ≈2^19.0, ε: 0.003906
```

Listing 1: Basic example of calling the LWE-Estimator of the LWE instance $n = 128$, $\alpha = \frac{1}{\sqrt{2\pi n \log^2 n}}$, $q \approx n^2$, and $m = 256$

Listing 2 shows the estimations of the LWE instance with $n = 128$, $\alpha = \frac{1}{\sqrt{2\pi n \log^2 n}}$, $q \approx n^2$, and $m = 256$

with the secret coefficients chosen uniformly random in $[-1, 0, 1]$.

```
sage: n,alpha,q = Param.Regev(128)
sage: m = 256
sage: costs=estimate_lwe(n,alpha,q,m=m,secret_distribution=(-1,1),skip="arora-gb,bkw,mitm")
usvp: rop: ≈2^37.9, m: 153, red: ≈2^37.9, δ_0: 1.011854, β: 54, d: 282, repeat: 44
dec: rop: ≈2^56.8, m: 235, red: ≈2^56.8, δ_0: 1.009308, β: 99, d: 363, babai: ≈2^42.2, babai_op: ≈2^57.3, repeat: 146, ε: 0.031250
dual: rop: ≈2^50.0, m: 369, red: ≈2^50.0, δ_0: 1.009552, β: 94, repeat: ≈2^16.6, d: 369, c: 14.400, k: 15, postprocess: 12
```

Listing 2: Example of calling the hardness estimations of the small secret LWE instance $n = 128$, $\alpha = \frac{1}{\sqrt{2\pi n \log^2 n}}$, $q \approx n^2$, $m = 256$, with secret coefficients chosen uniformly random in $\{-1, 0, 1\}$

In the following, we give interesting insights earned during the implementation.

One problem arises in the decoding attack `dec` when very strictly limiting the number of samples. It uses `enum_cost` to calculate the computational cost of the decoding step. For this, amongst other things, the stretching factors d_i of the parallelepiped are computed iteratively by step-wise increase as described in Section 4.2. In this process, the success probability is used, which is calculated as a product of terms $\text{erf}\left(\frac{d_i \|\mathbf{b}_i^*\| \sqrt{\pi}}{2\alpha q}\right)$, see Equation (7). Since the precision is limited, this may lead falsely to a success probability of 0. In this case, the loop never terminates. This problem can be avoided but doing so leads to an unacceptable long runtime. Since this case arises only when very few samples are given, our software throws an error, saying that there are too few samples.

The original LWE-Estimator routine to find the block size k for BKZ, called `k_chen`, iterates through possible values of k , starting at 40, until the resulting δ_0 is lower than the targeted δ_0 . As shown in Listing 3, this iteration uses steps of multiplying k by at most 2. When given a target- δ_0 close to 1, only a high value k can satisfy the used equation. Hence, it takes a long time to find the suitable k . Therefore, the previous implementation of finding k for BKZ is not suitable in case a limited number of samples is given. Thus, we replace this function in our implementation by finding k using the secant-method as presented in Listing 4.

```
f=lambda k:(k/(2*pi_r*e_r)*(pi_r*k)**(1/k))**
(1/(2*(k-1)))
while f(2*k) > delta:
    k *= 2
while f(k+10) > delta:
    k += 10
while True:
    if f(k) < delta:
        break
```

```
k += 1
```

Listing 3: Iteration to find k in method `k_chen` of the previous implementation used in the LWE-Estimator

```
f=lambda k: RR((k/(2*pi_r*e_r)*(pi_r*k)**(1/k))**
(1/(2*(k-1))-delta))
k=newton(f, 100, fprime=None, args=(), tol=1.48e-08,
maxiter=500)
k=ceil(k)
```

Listing 4: Implementation of method `k_chen` to find k using the secant-method

5.2 Comparison of Implementations and Algorithms

In the following, we present hardness estimations of LWE with and without taking a restricted number of samples into account. The presented experiments are done for the following LWE instance: $n = 128$, $\alpha = \frac{1}{\sqrt{2\pi n \log^2 n}}$, and $q \approx n^2$.

We show the base-2 logarithm of the estimated hardness of the LWE instance under all implemented attacks (except for Arora and Ge’s algorithm) in Table 15a and Table 15b. According to the experiments, the hardness decreases with increasing the number of samples and remains the same after reaching the optimal number of samples. If our software could not find a solution the entry is filled with NaN. This is mostly due to too few samples provided to apply the respective algorithm.

In Table 16, we show the logarithmic hardness and the corresponding optimal number of samples estimated for unlimited number of samples. It should be noted that some algorithms rely on multiple executions, e.g., to amplify a low success probability of a single run to a target success probability. In such a case, the previous implementation of the LWE-Estimator assumed new samples for each run of the algorithm. In our implementation, we assume that samples may be reused in repeated runs of the same algorithm, giving a lower bound on the hardness estimations. Hence, sometimes the optimal number of samples computed by the original LWE-Estimator and the optimal number of samples computed by our method differ a lot in Table 16, e.g., decoding attack. To compensate this and to provide better comparability, we *recalculate* the optimal number of samples.

Comparing Table 15 and Table 16 shows that for a number of samples lower than the optimal number of samples, the estimated hardness is either (much) larger than the estimation using optimal number of samples

| samples | mitm | dual | dec | usvp-primal | usvp-dual |
|---------|-------|-------|------|-------------|-----------|
| 100 | 326.5 | 127.3 | 92.3 | NaN | 95.7 |
| 150 | 326.5 | 87.1 | 65 | NaN | 55.7 |
| 200 | 326.5 | 77.2 | 57.7 | 263.4 | 49.2 |
| 250 | 326.5 | 74.7 | 56.8 | 68.8 | 48.9 |
| 300 | 326.5 | 74.7 | 56.8 | 51.4 | 48.9 |
| 350 | 326.5 | 74.7 | 56.8 | 48.9 | 48.9 |
| 400 | 326.5 | 74.7 | 56.8 | 48.9 | 48.9 |
| 450 | 326.5 | 74.7 | 56.8 | 48.9 | 48.9 |

(a)

| samples | bkw |
|-------------------|------|
| $1 \cdot 10^{21}$ | NaN |
| $2 \cdot 10^{21}$ | NaN |
| $4 \cdot 10^{21}$ | NaN |
| $6 \cdot 10^{21}$ | NaN |
| $8 \cdot 10^{21}$ | NaN |
| $1 \cdot 10^{22}$ | NaN |
| $2 \cdot 10^{22}$ | 85.1 |
| $4 \cdot 10^{22}$ | 85.1 |
| $6 \cdot 10^{22}$ | 85.1 |

(b)

Table 15: Logarithmic hardness of the algorithms exhaustive search (mitm), coded-BKW (bkw), distinguishing attack (sis), decoding (dec), standard embedding (usvp-primal), and dual embedding (usvp-dual) depending on the given number of samples for the LWE instance $n = 128$, $\alpha = \frac{1}{\sqrt{2\pi n \log^2 n}}$, and $q \approx n^2$

| Algorithm | Optimal number of samples | | hardness [bit] |
|-----------|---------------------------|-----------------------|----------------|
| | Original calculation | Recalculation | |
| mitm | 181 | 181 | 395.9 |
| sis | 192795128 | 376 | 74.7 |
| dec | 53436 | 366 | 58.1 |
| usvp | 16412 | 373 | 48.9 |
| bkw | $1.012 \cdot 10^{22}$ | $1.012 \cdot 10^{22}$ | 85.1 |

Table 16: Logarithmic hardness with optimal number of samples computed by the previous LWE-Estimator and the optimal number of samples recalculated according to the model used in this work for the LWE instance $n = 128$, $\alpha = \frac{1}{\sqrt{2\pi n \log^2 n}}$, and $q \approx n^2$

or does not exist. In contrast, for a number of samples greater or equal than the optimal number of samples, the hardness is exactly the same as in the optimal case, since the implementation falls back on the optimal number of samples when enough samples are given. Without this the hardness would increase again as can be seen for the dual embedding attack in Figure 2. For the results presented in Figure 2 we manually disabled the function to fall back to the optimal number of samples.

In Figure 3 we show the effect of limiting the available number of samples on the considered algorithms. We do

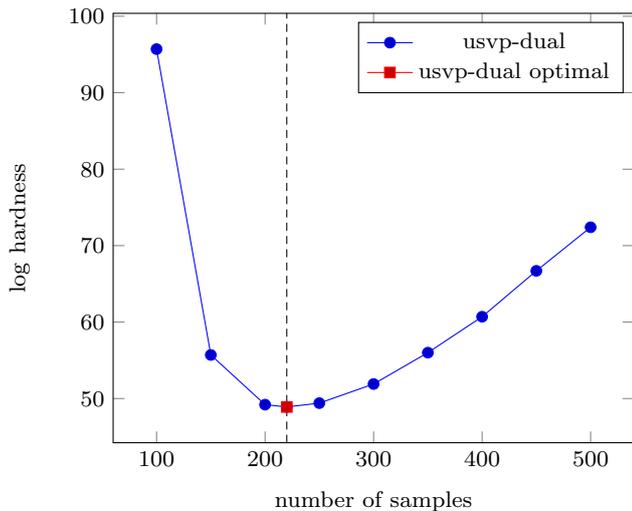


Fig. 2: Logarithmic hardness of dual embedding (usvp-dual) without falling back to optimal case for a number of samples larger than the optimal number of samples for the LWE instance $n = 128$, $\alpha = \frac{1}{\sqrt{2\pi n \log^2 n}}$, and $q \approx n^2$

not include coded-BKW in this plot, since the number of required samples to apply the attack is very large (about 10^{22}). The first thing that strikes is, that the limitation of the number of samples leads to an clearly notable increase of the logarithmic hardness for all shown algorithms but exhaustive search and BKW. The latter ones are basically not applicable for a limited number of samples. Furthermore, while the algorithms labeled with mitm, sis, dec, and usvp-primal are applicable for roughly the same interval of samples, the dual embedding algorithm (usvp-dual) stands out: the logarithmic hardness of dual-embedding is lower than for the other algorithms for $m > 150$. The reason is that during the dual embedding SVP is solved in a lattice of dimension $n + m$, when only m samples are given. Moreover, the dual embedding attack is the most efficient attack up to roughly 350 samples. Afterwards, it is as efficient as the standard embedding (usvp-primal).

6 Impact on Concrete Instances

We tested and compared various proposed parameters of different primitives such as signature schemes [5, 10], encryption schemes [24, 29], and key exchange protocols [6, 12, 13]. In this section we explain our findings using an instantiation of the encryption scheme by Lindner and Peikert [24] as an example. It aims at “medium security” (about 128 bits) and provides $n + \ell$ samples,

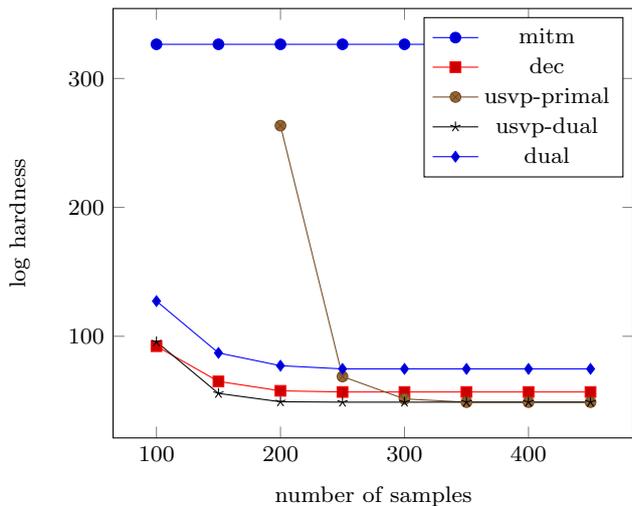


Fig. 3: Comparison of the logarithmic hardness of the LWE instance $n = 128$, $\alpha = \frac{1}{\sqrt{2\pi n \log^2 n}}$, and $q \approx n^2$ of the algorithms meet-in-the-middle (mitm), distinguishing (dual), decoding (dec), standard embedding (usvp-primal), and dual embedding (usvp-dual), when limiting the number of samples

where ℓ is the message length⁵. For our experiments, we use $\ell = 1$. The secret follows the error distribution, which means that it is not small. However, we expect a similar behavior for small secret instances.

Except bkw and mitm, all attacks considered use basis reduction as a subroutine. As explained in Section 3, several ways to predict the performance of basis reduction exist. Assuming that sieving scales as predicted to higher dimension leads to the smallest runtime estimates for BKZ on quantum (called $t_{k,q\text{-sieve}}$) and classical ($t_{k,\text{sieve}}$) computers. However, due to the subexponential memory requirement of sieving, it might be unrealistic that sieving is the most efficient attack (with respect to runtime and memory consumption) and hence enumeration might remain the best SVP solver even for high dimensions. Consequently, we include the runtime estimation of BKZ with enumeration ($t_{k,\text{enum}}$) to our experiments. Finally, we also performed experiments using the prediction by Lindner and Peikert (LP).

Our results are summarized in Table 17. We write “-” if the corresponding algorithm was not applicable for the tested instance of the Lindner-Peikert scheme. Since bkw and mitm do not use basis reduction as subroutine, their runtimes are independent of the used BKZ prediction.

⁵ The original publication states that $2n + \ell$ samples are provided, which is a minor mistake.

Table 17: Comparison of hardness estimations with or without accounting for restricted number of samples for the Linder-Peikert encryption scheme with $n = 256$, $q = 4093$, $\alpha = \frac{8.35}{q\sqrt{2\pi}}$, and $m = 384$

| LWE solver | $t_{k,q\text{-sieve}}$ | | $t_{k,\text{sieve}}$ | | $t_{k,\text{enum}}$ | | LP | |
|------------|------------------------|--------------|----------------------|--------------|---------------------|--------------|--------------|--------------|
| | $m = \infty$ | $m = 384$ | $m = \infty$ | $m = 384$ | $m = \infty$ | $m = 384$ | $m = \infty$ | $m = 384$ |
| mitm | 407.0 | 407.0 | 407.0 | 407.0 | 407.0 | 407.0 | 407.0 | 407.0 |
| usvp | 97.7 | 102.0 | 104.2 | 108.9 | 144.6 | 157.0 | 149.9 | 159.9 |
| dec | 106.1 | 111.5 | 111.5 | 117.2 | 138.0 | 143.4 | 144.3 | 148.7 |
| dual | 106.2 | 132.5 | 112.3 | 133.1 | 166.0 | 189.1 | 158.0 | 165.2 |
| bkw | 212.8 | - | 212.8 | - | 212.8 | - | 212.8 | - |

For the LWE instance considered, the best attack with arbitrary many samples always remains the best attack after restricting the number of samples. Restricting the samples always leads to an increased runtime for every attack, up to a factor of 2^{26} . Considering only the best attack shows that the hardness increases by about 5 bits. Unsurprisingly, usvp (which consists nearly solely of basis reduction) performs best when we assume that BKZ is fast, but gets outperformed by the decoding attack when we assume larger runtimes for BKZ.

7 Summary

In this work, we present an analysis of the hardness of LWE for the case of a restricted number of samples. For this, we describe the approaches distinguishing attack, decoding, standard embedding, and dual embedding shortly and analyze them with regard to a restricted number of samples. Also, we analyze the small secret variants of the mentioned algorithms under the same restriction of samples.

We adapt the existing software tool *LWE-Estimator* to take the results of our analysis into account. Moreover, we also adapt the algorithms BKW and meet-in-the-middle that are omitted in the theoretical description. Finally, we present examples, compare hardness estimations with optimal and restricted numbers of samples, and discuss our results.

The usage of a restricted set of samples has its limitations, e.g., if given too few samples, attacks are not applicable as in the case of BKW. On the other hand, it is possible to construct LWE instances from a given set of samples. For example, in [16] ideas how to generate additional samples (at cost of having higher noise) are presented. An integration in the LWE-Estimator and comparison of those methods would give an interesting insight, since it may lead to improvements of the estimation, especially for the algorithms exhaustive search and BKW.

References

1. Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the BKW algorithm on LWE. *Designs, Codes and Cryptography*, 74, 2015.
2. Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-svp. In *Information Security and Cryptology - ICISC 2013*, volume 8565 of *LNCS*, 2013.
3. Martin R. Albrecht, Florian Göpfert, Cedric Lefebvre, Rachel Player, and Sam Scott. Estimator for the bit security of LWE instances. <https://bitbucket.org/malb/lwe-estimator>, 2016. [Online; accessed 01-June-2017].
4. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3), 2015.
5. Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. Revisiting TESLA in the quantum random oracle model. In *PQCrypto 2017*, *LNCS*. Springer, 2017.
6. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In *USENIX Security Symposium*, 2016.
7. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO 2009*, volume 5677 of *LNCS*. Springer, 2009.
8. Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *ICALP 2011*, volume 6755 of *LNCS*. Springer, 2011.
9. László Babai. On lovász' lattice reduction and the nearest lattice point problem. In K. Mehlhorn, editor, *STACS 1985*. Springer, 1985.
10. Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *Topics in Cryptology - CT-RSA 2014*, volume 8366 of *LNCS*, 2014.
11. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *SODA 2016*. SIAM, 2016.
12. Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *CCS 2016*. ACM, 2016.
13. Joppe Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *IEEE Symposium on Security and Privacy (S&P) 2015*, 2015.
14. Yuanmi Chen and Phong Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT 2011*, volume 7073 of *LNCS*. Springer, 2011.

15. Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist.*, 23(4):493–507, 12 1952.
16. Alexandre Duc, Florian Tramèr, and Serge Vaudenay. Better algorithms for LWE and LWR. In *EUROCRYPT 2015*, volume 9056 of *LNCS*. Springer, 2015.
17. Nicolas Gama, Phong Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, 2010.
18. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC '08*. ACM, 2008.
19. Florian Göpfert. *Securely Instantiating Cryptographic Schemes Based on the Learning with Errors Assumption*. PhD thesis, Darmstadt University of Technology, Germany, 2016.
20. Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Algorithms for the shortest and closest lattice vector problems. In *IWCC 2011*. Springer, 2011.
21. H. W. Lenstra jr. Integer programming with a fixed number of variables. *MATH. OPER. RES*, 8(4):538–548, 1983.
22. Thijs Laarhoven, Michele Mosca, and Joop van de Pol. Solving the shortest vector problem in lattices faster using quantum search. In *PQCrypto 2013*, volume 7932 of *LNCS*. Springer, 2013.
23. A.K. Lenstra, H.W. Lenstra jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.
24. Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *Topics in Cryptology - CT-RSA 2011*, volume 6558 of *LNCS*, 2011.
25. Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *CRYPTO 2009*, volume 5677 of *LNCS*. Springer, 2009.
26. Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*. Springer, 2009.
27. Phong Q. Nguyen and Damien Stehlé. Floating-point LLL revisited. In *EUROCRYPT 2005*, volume 3494 of *LNCS*. Springer, 2005.
28. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. *STOC '09*. ACM, 2009.
29. El Bansarkhani Rachid. Lara - a design concept for lattice-based encryption. *Cryptology ePrint Archive*, Report 2017/049, 2017.
30. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC 2005*. ACM, 2005.
31. C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66(1):181–199, 1994.
32. Özgür Dagdelen, Rachid El Bansarkhani, Florian Göpfert, Tim Güneysu, Tobias Oder, Thomas Pöppelmann, Ana Helena Sánchez, and Peter Schwabe. High-speed signatures from standard lattices. In *LATINCRYPT 2014*, volume 8895. Springer, 2015.