Guidance for Efficient Selection of Secure Parameters for Fully Homomorphic Encryption

Elena Kirshanova[1], Chiara Marcolla[1], and Sergi Rovira[2]

[1] Technology Innovation Institute, Abu Dhabi, United Arab Emirates
[2] Pompeu Fabra University, Barcelona, Spain

Abstract. The field of Fully Homomorphic Encryption (FHE) has seen many theoretical and computational advances in recent years, bringing the technology closer to practicality than ever before. For this reason, practitioners from neighbouring fields such as machine learning have sought to understand FHE to provide privacy to their work. Unfortunately, selecting secure and efficient parameters in FHE is a daunting task due to the many interdependencies between the parameters involved. In this work, we solve this problem by moving away from the standard parameter selection procedure, introducing formulas which provide secure and optimal parameters for any lattice-based scheme. We build our formulas from a strong theoretical foundation based on cryptanalysis against LWE.

Keywords: Fully Homomorphic Encryption, Parameter Selection, Learning With Errors, Primal attacks, Bounded Distance Decoding

1 Introduction

With the advancements of future-generation networking technologies like cloud services, artificial intelligence applications, Internet of Things, and edge computing, concerns about data privacy are increasing significantly. Homomorphic encryption serves as a solution for preserving privacy during data processing, allowing computations on encrypted data without the need for decryption. More specifically, Fully Homomorphic Encryption (FHE) schemes define ciphertext operations corresponding to computations on the underlying plaintext as additions or multiplications [1, 16, 44, 45].

The first FHE scheme was introduced in 2009 by Gentry [32]. Gentry provided a method for constructing a general FHE scheme from a scheme with limited but sufficient homomorphic evaluation capacity. Since then, several FHE constructions have been proposed, being BGV [14], BFV [13, 30], TFHE [19, 20], and CKKS [17, 18], the most well-known schemes in the field.

The security of most FHE schemes is based on the presumed intractability of the (decision) Learning with Errors (LWE) problem, [49], and its ring variant (RLWE), [42]; the latter version is often preferred for efficiency reasons. Informally, decisional LWE consists in distinguishing equations $\{(a_i, b_i = s \cdot a_i + e_i)\}_i$

mod q, perturbed by small noise e_i (also called error), from uniform random tuples from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.[3] To guarantee a correct decryption, the noise added has to be small. Indeed, roughly speaking, if the error surpasses the modulus q, it can result in a wrap-around effect, altering the decrypted output. Therefore, maintaining a limited noise level and computing a tight bound of the error's size is essential for ensuring the recovery of the plaintext.

The problem arising from lattice-based constructions is that the noise grows whenever an homomorphic operation is performed. In particular, it grows exponentially when homomorphic multiplications are computed. To increase the number of supported operations, we could increase the ciphertext modulus q. However, a larger modulus also decreases the security level of the underlying scheme, requiring a larger LWE dimension n to keep the same security level, which comes at the cost of efficiency.

This required trade-off between security (small q) and error margin (large q) illustrates the challenge of identifying an optimal set of parameters for a given FHE scheme. Such a balancing process, called *parameter estimation*, is one of the main issues that need to be addressed to make FHE practical.

In this work, we deduce precise and closed-form formulas for parameter selection of LWE-based schemes. Our approach to parameter selection greatly simplifies the development and deployment of FHE-based privacy-preserving applications. That is, we provide a fast methodology to easily choose secure parameters without the need to understand the underlying cryptanalytic details or the interdependencies between the parameters.

Related works. Several efforts have been made by the FHE community to address the challenge of facilitating the deployment of FHE among researchers and practitioners and to select an optimal set of parameters. For instance, some FHE compilers, which are high-level tools that aim at abstracting the technical APIs exposed by FHE libraries, allow a sort of automatic parameter generation (for *specific* FHE schemes) according to some predefined requirements [44, 55]. Some examples are ALCHEMY [23], Cingulata [15], EVA [25] and SEALion [29]. For more details of supported schemes and the methodology behind automatic parameter selection in FHE compilers, we refer the reader to [55].

Moreover, Bergerat *et al.* [10] proposed a framework for efficiently selecting parameters in TFHE-like schemes. Mono *et al.* [47] developed an interactive parameter generator for the levelled BGV scheme that supports arbitrary circuit models and Biasioli *et al.* [11] extended [47] to the BFV scheme.

Finally, important steps have been made to address parameter selection, providing methods and formulas for *any* FHE schemes. For instance, the Homomorphic Encryption Standard [3] provides upper limits on the size of the ciphertext modulus for certain security levels λ and dimension n in the form of lookup

[3] While in FHE literature n is often referred to as polynomial degree, having in mind Ring-LWE based constructions, in this work we refer to n as to LWE dimension, as we do not utilize any algebraic properties of Ring-LWE.

tables, using the Lattice Estimator[4] [5]. Moreover, in [47], the authors proposed a compact formula that computes the hardness of LWE for given dimension n, modulus q, the standard deviation of secret distribution σ_s, and σ_e – the standard deviation of the error distribution. Yet, as emphasised in the invited talk of Paillier [48], these efforts are still insufficient to provide an efficient easy-to-use tool for secure parameter selection.

Our contribution. In this paper, we present a novel method for determining optimal parameters for any FHE scheme, focusing on the *macro* level, namely n, q, σ_s, σ_e and λ. To achieve this, starting with the theoretical foundation of lattice attacks, we introduce formulas that select these macro parameters. Concretely, our contributions are the following.

1. We consider the most relevant lattice attacks (for FHE parameters) and derive closed and precise formulas of their running times as functions of n, q, σ, and λ.
2. These formulas enable us to express the LWE dimension n as a function of $\lambda, q, \sigma_e, \sigma_s$, giving a way to choose, for a desired security level λ, an appropriate n. We verify our formulas by running extensive experiments with the Lattice Estimator [5]. By creating a large dataset of points that relates n, q, σ, λ, we are able to fine-tune our formulas to make sure that lower-order terms in the derived expressions are of the correct form and, hence, provide accurate estimates for broad parameter sets.
3. We provide Python scripts that implement these formulas, and we specify the best practices for using them. The scripts are publicly available on our Github repository[5].

Our analysis considers two types of lattice algorithms: the so-called bounded distance decoding attack and the unique Shortest Vector Problem attack. We chose these two as currently they are the most efficient attacks whose correctness was not refuted. Efficient dual attacks that can outperform the attacks we consider here, at the time of writing, do not offer correctness [27].

Comparison with related work. In [10], the authors build a framework to efficiently find optimal parameters for TFHE-like schemes. Their methodology relies on a *security oracle*, which given the parameters n, q, λ and σ_s, outputs the minimal σ_e that guarantees security λ. In practice, this oracle is constructed as a linear approximation. Our methodology deviates considerably from their approach. The main difference is that our formulas do not come solely from empirical results but from the analysis of the main lattice attacks. The point of contact of the two works is the use of the Lattice Estimator to build a database and the use of a fitting function. However, while [10] uses the fitting function to

[4] The Lattice Estimator (`https://github.com/malb/lattice-estimator`) is the successor of the LWE Estimator, which is a software tool to determine the security level of LWE instances under various attacks proposed until the present time.

[5] https://github.com/sergirovira/fastparameterselection

build the totality of their formula, our use is solely for optimizing lower-order terms.

We want to highlight that our formulas provide not only an alternative to the existing procedures of parameter selection in FHE but also a faster paradigm. That is, using a script-based strategy (such as running the Lattice Estimator for different sets of parameters) is inefficient since the only way to obtain suitable parameters is trial and error, which can mean checking hundreds of cases until the optimal parameters are found. Using a look-up table of pre-computed values is, of course, faster but also limited since it might not accommodate all possible needs that arise when selecting parameters for FHE schemes. This approach is used in the vast majority of FHE libraries [8, 40, 53]. Using a formula-based method, we get the best of both approaches. Namely, we can get optimal parameters for any given application instantly. Another advantage of using formulas is that we can understand the behaviour of the parameters in relation to each other, allowing us to easily check if the parameters we are using are optimal. Finally, it is worth mentioning that our formulas are applicable to *any* construction based on the hardness of LWE and not only to FHE schemes.

To conclude, our approach significantly accelerates the parameter selection process, offering a practical and efficient tool for researchers and practitioners deploying FHE in real-world applications.

2 Preliminaries

2.1 Notation

For a positive integer q, we denote by $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ the ring of integers modulo q. For $n \geq 1$, denote by $\mathbb{R}^n$ the real vector space. For a vector $\mathbf{x}$, both $\mathbf{x}_i$ and $\mathbf{x}[i]$ denote either the i-th scalar component of the vector or the i-th element of an ordered finite set of vectors. Matrices are denoted by bold capital letters. We denote by $\|\mathbf{x}\|$ the Euclidean norm of $\mathbf{x}$. By $\mathbf{A}^t$ we denote the transpose of $\mathbf{A}$.

2.2 Mathematical background

Let $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_k)$ be linearly independent vectors in $\mathbb{R}^n$, then we can define the *lattice* $\mathcal{L}(\mathbf{B})$ generated by $\mathbf{B}$ as the set of all integer linear combinations of elements of $\mathbf{B}$:

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) = \Big\{ \sum_{i=1}^{k} \gamma_i \mathbf{b}_i : \quad \gamma_i \in \mathbb{Z}, \mathbf{b}_i \in \mathbf{B} \Big\}.$$

If $k = n$, the lattice is said to be *full rank*. We will be concerned with integral lattice, i.e., $\mathcal{L} \subset \mathbb{Z}^n$. An integral lattice $\mathcal{L}$ is called q-ary if $q\mathbb{Z}^n \subset \mathcal{L} \subset \mathbb{Z}^n$. The determinant of a lattice $\mathcal{L}$ defined by a basis $\mathbf{B}$ is $\det(\mathcal{L}) = \sqrt{\det(\mathbf{B}^t\mathbf{B})}$ and is independent of the choice of basis.

For basis vectors $\mathbf{b}_i$, we write $\mathbf{b}_i^\star$ for the corresponding Gram-Schmidt vectors. Concretely, the i-th Gram-Schmidt vector $\mathbf{b}_i^\star$ is the projection of $\mathbf{b}_i$ orthogonally to the subspace $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$. We denote such projecting operator π_i. We write $\mathbf{B}_{[i,j]}$ to denote the matrix whose columns are $\{\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j)\}$. It generates (a projective) sublattice of dimension $j - i + 1$. We will make use of the fact that $\det(\mathcal{L}(\mathbf{B})) = \prod_{i=1}^n \|\mathbf{b}_i^\star\|$.

The *minimum distance* or the *first successive minimum* of lattice $\mathcal{L}$, denoted by $\lambda_1(\mathcal{L})$, is the Euclidean norm of a shortest non-zero vector in $\mathcal{L}$: $\lambda_1(\mathcal{L}) = \min\{\|\mathbf{v}\| : \mathbf{v} \in \mathcal{L}, \mathbf{v} \neq 0\}$. The i-th successive minimum $\lambda_i(\mathcal{L})$ is the smallest $r > 0$ such that $\mathcal{B}(\mathbf{0}, r)$ contains i linearly independent vectors of $\mathcal{L}$, where $\mathcal{B}(\mathbf{0}, r)$ is a ball in $\mathbb{R}^n$ of radius r centered at $\mathbf{0}$. The successive minima are independent of the basis choice.

The *Gaussian Heuristic* predicts $\lambda_1(\mathcal{L})$ for an n-dimensional lattice $\mathcal{L}$:

$$\lambda_1(\mathcal{L}) \approx \frac{\sqrt{n}}{\sqrt{2\pi e}} (\det(\mathcal{L}))^{1/n}.$$

Hard problems on lattices. There are several fundamental problems related to lattices, the following ones are relevant to this work.

The *Shortest Vector Problem (SVP)* asks to find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$.

In the promise variant of SVP, the so-called *unique SVP (uSVP)*, we are guaranteed that the first successive minimum is $\gamma > 1$ times smaller than the second minimum λ_2. We are asked to find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$.

The *Closest Vector Problem (CVP)* asks to find $\mathbf{v} \in \mathcal{L}$ closest to a given target vector $\mathbf{t} \in \mathbb{R}^n$.

Given a lattice $\mathcal{L}$ and a target vector $\mathbf{t}$ close to the lattice, the *Bounded Distance Decoding (BDD)* problem asks to find $\mathbf{v} \in \mathcal{L}$ closest to the target $\mathbf{t}$ with the promise that $\|\mathbf{t} - \mathbf{v}\| \leq R$, where $R \ll \lambda_1(\mathcal{L})$.

Discrete Gaussian Distribution. For a vector $\mathbf{v}$ and any $\sigma > 0$, define $\rho_\sigma(\mathbf{v}) = \exp(-\pi\|\mathbf{v}\|^2/(2\pi\sigma^2))$. For a lattice $\mathcal{L}$, the *discrete Gaussian probability distribution* with standard deviation σ[6] is defined with the probability density function

$$\mathcal{D}_{\mathcal{L},\sigma}(\mathbf{v}) = \frac{\rho_\sigma(\mathbf{v})}{\sum_{\mathbf{x}\in\mathcal{L}} \rho_\sigma(\mathbf{x})}.$$

2.3 Lattice reduction

Lattice reduction aims at improving the quality of a lattice basis. In this work, we are interested in the lattice reduction algorithm called BKZ (short for Block-Korkine-Zolotarev, [50]). Together with a lattice basis, it receives as input an

[6] Notice that the variance of a Discrete Gaussian and a Continuous Gaussian does not match when $\sigma \leq 0.6$. In this paper we use the same parameter for both since we always work with $\sigma > 0.6$.

integer parameter β (called the *block size*) that governs the quality of the output basis and the runtime. Here by 'quality' we mean the Euclidean norm of the shortest vector in the basis output by BKZ. Concretely, BKZ run with block size β on a lattice $\mathcal{L}$ of rank n, returns a basis containing a lattice vector $\mathbf{b}_1$ of norm

$$\|\mathbf{b}_1\| = \delta_\beta^n \cdot \det(\mathcal{L})^{1/n}, \tag{1}$$

where δ_β is known as the root Hermite-factor and can be expressed in terms of β as

$$\delta_\beta = (((\pi\beta)^{1/\beta}\beta)/(2\pi e))^{\frac{1}{2(\beta-1)}} \approx \left(\frac{\beta}{2\pi e}\right)^{\frac{1}{2\beta}}, \tag{2}$$

where the approximation holds for large β's such that $(\pi\beta)^{1/\beta} \approx 1$.

The BKZ-β algorithm works by calling multiple times an algorithm for SVP on sublattices of dimension β. In [34] it is shown that after poly(n) many number of SVP calls, the guarantee defined in Equation (1) is achieved. Hence, the running time of BKZ is determined by the complexity of SVP in β dimensional lattices. The asymptotically fastest algorithm for SVP is due to Becker-Gama-Ducas-Laarhoven [9] that outputs a shortest vector in an n-dimensional lattice in time $2^{0.292n+o(n)}$. We choose this running time (ignoring the $o()$-term) as the measure of SVP hardness. Further, for a more concrete complexity of BKZ-β on an n-dimensional lattice we set the running time of BKZ as

$$T_{\mathrm{BKZ}}(\beta, n) = 2^{0.292\beta} \cdot 8n \cdot 16.4, \tag{3}$$

which is the choice adopted by [12, 28, 31]. The correcting constant of 16.4 obtained experimentally [9]. The concrete choice of $T_{\mathrm{BKZ}}(\beta, n)$ is called the core-SVP model [6]. Our results are easy to adapt to other existing choices of $T_{\mathrm{BKZ}}(\beta, n)$.

In addition to Equation (1), BKZ quality guarantees extend (heuristically) to norms of Gram-Schmidt vectors of the returned basis. It is formulated in Geometric Series Assumption. All known lattice estimators [5, 24] rely on this assumption.

Definition 1 (Geometric Series Assumption (GSA), [51]). *The norms of Gram-Schmidt vectors of a BKZ-β reduced basis satisfy*

$$\|\mathbf{b}_i^\star\| = \alpha^{i-1}\|\mathbf{b}_1\|,$$

where $\alpha = \delta_\beta^{\frac{-2n}{n-1}} \approx \delta_\beta^{-2} \approx \beta^{-1/\beta}$.

Babai's algorithm. For one of the attacks considered in this work, we need an efficient BDD solver: Babai's algorithm [7]. Its running time is polynomial in the lattice dimension. In a BDD instance, we are given a lattice basis $\mathbf{B}$ and the target $\mathbf{t}$. Assume for simplicity that the coordinates of $\mathbf{t}$ are independent Gaussians with standard deviation σ (case of LWE). Informally, the success probability of Babai depends on the relation between $\|\mathbf{b}_i^\star\|$ and σ: if $\|\mathbf{b}_n^\star\| > \sigma$, the success probability is constant, while if $\|\mathbf{b}_1^\star\| = \sigma$, the success probability is super-exponentially low (in the lattice dimension). We will be concerned with the first case (constant success probability) formally defined in the next claim. We use the formulation from [35].[7]

Lemma 1 ([35, Lemma 4]). *Let the sequence $\|\mathbf{b}_1^\star\|, \dots, \|\mathbf{b}_n^\star\|$ follow GSA, and let $\mathbf{t}$ be a vector with coordinates distributed as independent Gaussians with standard deviation σ. The success probability of Babai's algorithm is $1 - o(1)$, if $\|\mathbf{b}_n^\star\| > \sigma(\log n)^{1/2+\varepsilon}$ for fixed constant $\varepsilon > 0$.*

2.4 The Learning With Errors Problem

The Learning with Errors problem (LWE) was introduced by Regev in [49]. The LWE problem is parametrized by an integer n, modulus q (not necessarily prime), an error distribution $\chi_e : \mathbb{Z}_q \to \mathbb{R}^+$ with standard deviation σ_e, and a secret distribution $\chi_s : \mathbb{Z}_q \to \mathbb{R}^+$ with standard deviation σ_s.

Definition 2 (The Learning with Errors (LWE) problem). *Given a vector $\mathbf{b} \in \mathbb{Z}_q^m$ and a matrix $\mathbf{A}$ taken uniformly at random from $\mathbb{Z}_q^{m\times n}$, the search version of the LWE problem consists in finding an unknown vector $\mathbf{s} \in \mathbb{Z}_q^n$ such that*

$$\mathbf{A}\,\mathbf{s} + \mathbf{e} = \mathbf{b} \bmod q,$$

where $\mathbf{e} \in \mathbb{Z}_q^m$ is sampled coordinate-wise from an error distribution χ_e, and $\mathbf{s}$ is sampled coordinate-wise from χ_s. In other words, the goal is to find a vector $\mathbf{s} \in \mathbb{Z}_q^n$ given a list of m noisy equations from

$$\mathcal{A}_{\mathbf{s},\chi_e,\chi_s} = \{(\mathbf{a}_i, b_i = \langle \mathbf{a}_i, \mathbf{s}\rangle + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q : \mathbf{a}_i \overset{\$}{\leftarrow} \mathbb{Z}_q^n,\, e_i \leftarrow \chi_e, \mathbf{s}_i \leftarrow \chi_s\}.$$

Often in FHE constructions, we have $\chi_s \in \{\mathcal{U}_3, \mathcal{U}_2\}$, the uniform distribution on $\mathbb{Z}_3$ (called *ternary secret* LWE) or on $\mathbb{Z}_2$ called (*binary secret* LWE). For the error, we are concerned with discrete Gaussian distribution centered in 0 with standard deviation $\sigma_e = 3.19$ [3].

There exist several versions of LWE: Ring-LWE [42, 54] and Module-LWE [39]. These are mainly used for efficiency reasons, security-wise these versions, at the time of writing, are believed to be equivalent to 'plain' LWE. Therefore, all our results extend to these other versions, in particular to Ring-LWE, the most relevant variant in the FHE context.

3 Deriving LWE dimension for required security level

On chosen algorithms. We focus on *primal* attacks on LWE, and do not consider the so-called dual attacks. First, the recent discoveries [27] of failing heuristics

[7] Even though in [35, Lemma 4] the authors talk about *continuous* Gaussian, the result holds for the discrete Gaussian too.

employed in efficient dual attacks [33, 46] invalidate the claimed complexities. Despite of ongoing attempts to bring dual attacks back into play [26], no complete algorithm is presented that outperforms primal attacks. While other potentially less efficient versions of dual attacks have not been invalidated, the primal attacks perform better on the parameters considered in this work. Second, dual attacks seem to be much harder to implement: we are not aware of an existing implementation of a competitive dual attack.

We neither consider here the so-called *hybrid* attacks [2, 36]. These are relevant for sparse secret LWE, i.e., for cases when the Hamming weight of the secret is less than $n/2$. The analysis of these attacks is left for future work.

We receive on input an LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$, where $\mathbf{s}$ follows the distribution χ_s with standard deviation σ_s, and $\mathbf{e}$ follows the distribution χ_e with standard deviation σ_e. We now describe in details the two attacks: BDD and uSVP, derive accurate formulas for their complexities, and finally reverse these formulas to express n as a function of q, σ_e, σ_s, and the desired security level λ.

3.1 The BDD attack

While the BDD attack on LWE has been known for years [41], we did not find a reference that aligns well the Lattice Estimator [5], hence we first describe the attack, then derive its running time and reverse the runtime expression for the desired parameters, e.g., the LWE dimension n.

The search LWE problem is an average-case BDD problem for the $(m+n)-$dimensional q-ary lattice

$$\mathcal{L}_{\mathsf{bdd}} = \{\mathbf{v} \in \mathbb{Z}^{n+m} \mid [\mathbf{A}|\mathbf{I}_m]\mathbf{v} = 0 \bmod q\},$$

with the target vector $(\mathbf{0}, \mathbf{b}) \in \mathbb{Z}^n \times \mathbb{Z}^m$. To see this, consider a basis for this lattice over $\mathbb{Z}^{m+n}$ given by the columns of the matrix

$$\mathbf{B}_{\mathsf{bdd}} = \begin{pmatrix} \mathbf{I}_n & 0 \\ \mathbf{A} & q\mathbf{I}_m \end{pmatrix}.$$

From the LWE equation $\mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{b} - \mathbf{k} \cdot q$ for some $\mathbf{k} \in \mathbb{Z}^m$, we know that

$$\mathbf{B}_{\mathsf{bdd}} \cdot (\mathbf{s}, \mathbf{k})^t = (\mathbf{s}, \mathbf{b} - \mathbf{e})^t = (\mathbf{s}, -\mathbf{e})^t + (\mathbf{0}, \mathbf{b})^t.$$

The lattice $\mathcal{L}_{\mathsf{bdd}}$ is with high probability of full rank $m+n$ (since $\mathbf{A}$ has full column rank n with high probability) and the determinant of $\mathcal{L}_{\mathsf{bdd}}$ is $\det(\mathcal{L}_{\mathsf{bdd}}) = q^m$. The Gaussian Heuristic suggests that

$$\lambda_1(\mathcal{L}_{\mathsf{bdd}}) \approx \frac{\sqrt{m+n}}{2\pi e} \cdot q^{\frac{m}{m+n}}.$$

Further, the vector $(\mathbf{0}, \mathbf{b})$ is at distance $\|(\mathbf{s}, \mathbf{e})\| \approx \sqrt{n\sigma_s^2 + m\sigma_e^2} \ll \lambda_1(\mathcal{L}_{\mathsf{bdd}})$ from $\mathcal{L}_{\mathsf{bdd}}$, hence we have a BDD instance $(\mathcal{L}_{\mathsf{bdd}}, (\mathbf{0}, \mathbf{b}))$.

In cases were $\sigma_s < \sigma_e$, one can 're-balance' the contribution of $\mathbf{s}, \mathbf{e}$ into the distance $\sqrt{n\sigma_s^s + m\sigma_e^2}$ by scaling the $\mathbf{I}_n$ part of $B_{\mathtt{bdd}}$ by $\zeta = \max\{1, \lfloor \sigma_e/\sigma_s \rceil\}$, that is we perform the attack on $\mathbf{B}_{\mathtt{bdd}} = \begin{pmatrix} \zeta\mathbf{I}_n & 0 \\ \mathbf{A} & q\mathbf{I}_m \end{pmatrix}$. Even though it increases the distance of the target to the lattice, it also scales $\det(\mathcal{L}_{\mathtt{bdd}})$ by a factor ζ^n, which in turn increases $\lambda_1(\mathcal{L}_{\mathtt{bdd}})$ and hence the decoding properties of $\mathcal{L}_{\mathtt{bdd}}$. For FHE parameters, the secret $\mathbf{s}$ is often binary or ternary, in which cases $\zeta = \sigma_e/(1/2) = 2\sigma_e$ or $\zeta = \sigma_e/(\sqrt{2/3}) = \sqrt{3/2}\sigma_e$.

Denote for simplicity $d := m + n$, the dimension of $\mathbf{B}_{\mathtt{bdd}}$. The bounded distance decoding algorithm [41] works in three steps. In Step 1, we run a BKZ-β lattice reduction algorithm on $\mathbf{B}_{\mathtt{bdd}}$. Denote the output basis by $\mathbf{B}'_{\mathtt{bdd}}$. The goal of BKZ is to obtain a basis with the property

$$\lambda_1(\mathbf{B}'_{\mathtt{bdd},[d-\eta,d]}) < \|\pi_{d-\eta}((\mathbf{s}, \mathbf{e}))\|$$

for $0 \leq \eta < d$ as small as possible. Under the Gaussian Heuristic and the approximation $\|\pi_{d-\eta}((\mathbf{s}, \mathbf{e}))\| \approx \sigma_e\sqrt{\eta}$, the above inequality can be rewritten as

$$\frac{\sqrt{d}}{2\pi e} \det\left(\mathbf{B}'_{\mathtt{bdd},[d-\eta+1,d]}\right)^{1/d} < \sigma_e\sqrt{\eta}. \tag{4}$$

This condition means that the orthogonal projection of our short vector $(\mathbf{s}, \mathbf{e})$ on $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1, \ldots, \mathbf{b}_{d-\eta+1})$ is shorter than the shortest vector in the projected lattice $\mathbf{B}'_{\mathtt{bdd},[d-\eta+1,d]}$ given by the basis $(\pi_{d-\eta+1}(\mathbf{b}'_{d-\eta+1}), \ldots, \pi_{d-\eta+1}(\mathbf{b}'_d))$. In the LWE setting, GSA suggests that for small η's the left-hand side of Ineq. (4) is always larger than the right-hand side. Although both sides decrease for decreasing η, the left-hand side does it faster (again, due to GSA) and at some point Ineq. (4) is satisfied.

This implies that running an SVP solver on $[\mathbf{B}'_{\mathtt{bdd},[d-\eta+1,d]} | \pi_{d-\eta+1}((\mathbf{0}, \mathbf{b}))]$ will find the *projection* $\pi_{d-\eta+1}((\mathbf{s}, \mathbf{e}))$ of our secret. This SVP call constitutes the second step of the algorithm. Notice that we call SVP on a rank-(η) lattice generated by $[\mathbf{B}'_{\mathtt{bdd},[d-\eta+1,d]} | \pi_{d-\eta+1}((\mathbf{s}, \mathbf{e}))]$.

The third step of the attack 'lifts' the found projected vector $\pi_{d-\eta+1}((\mathbf{s}, \mathbf{e}))$ using Babai's algorithm on the 'remaining' part of the lattice $\mathbf{B}'_{\mathtt{bdd},[1,d-\eta+1]}$, which is a sublattice of $\mathbf{B}'_{\mathtt{bdd}}$ generated by its first $(d-\eta+1)$ vectors. The norms of Gram-Schmidt vectors of this sublattice, $\|\mathbf{b}_1^\star\|, \ldots \|\mathbf{b}_{d-\eta+1}^\star\|$ satisfy

$$\|\mathbf{b}_i^\star\| \geq \lambda_1(\mathbf{B}'_{\mathtt{bdd},[i,d]}) \geq \sigma_e\sqrt{d-i}, \quad i \leq d - \eta + 1,$$

where the first inequality comes from the fact that $\mathbf{b}_i^\star \in B'_{\mathtt{bdd},[i,d]}$, and the second is due to Ineq. (4). Applying Lemma 1 to $\mathbf{B}'_{\mathtt{bdd},[1,d-\eta+1]}$ gives constant probability of Babai algorithm to output $(\mathbf{s}, \mathbf{e})$.

Runtime analysis of BDD. Let us now analyse the runtime of this attack. Among the three steps of the BDD attack, the most expensive ones are the first step

(BKZ-β) and the second (SVP in dimension η). It is optimal to balance these two steps.

The runtime of BKZ-β on a d-dimensional lattice as given in Equation (3) is $T_{\mathrm{BKZ}}(\beta, d) = 2^{0.292\beta} \cdot 8d$, while the runtime of SVP on η-dimensional lattice is $T_{\mathrm{SVP}}(\eta) = 2^{0.292\eta}$. The two runtimes differ only by the $\log(8d)$ factor, hence we expect $\beta \approx \eta$ to be optimal. Indeed, running the estimator confirms this choice.

The required β can be derived from Ineq. (4). Concretely, using GSA and the BKZ-β guarantee on $\|\mathbf{b}_1'\|$, we compute

$$\det\left(\mathbf{B}'_{\mathsf{bdd},[d-\eta+1,d]}\right) = \prod_{i=d-\eta+1}^{d} \|\mathbf{b}_i^\star\| = \prod_{i=d-\eta+1}^{d} \delta_\beta^{d-1-2i} (\det \mathbf{B}_{\mathsf{bdd}})^{\frac{1}{d}} = \delta_\beta^{-\eta(d-\eta+2)} \cdot (q^m \zeta^n)^{\frac{\eta}{d}}.$$

From now on we use the approximation $\beta \approx \eta$ and work with β only. Here we notice that in LWE one is free to choose the number of samples m, which in turn affects the lattice dimension d. Minimizing the expression $\delta_\beta^{-\eta(d-\eta+2)} \cdot (q^m \zeta^n)^{\frac{\eta}{d}}$ with respect to d, yields optimal lattice dimension $d = \sqrt{\frac{n \ln(q/\zeta)}{\ln \delta_\beta}}$. From Ineq. (4) and Equation (2), we obtain the following expression for β as a function of d, q, σ_e, ζ:

$$\beta \geq \frac{d \ln\left(\frac{\beta}{2\pi e}\right)}{\ln\left(\frac{\beta}{2\pi e}\right) + 2\ln\left(\frac{q}{\sigma_e\sqrt{2\pi e}}\right) - 2\frac{n}{d}\ln\left(\frac{q}{\zeta}\right)}. \tag{5}$$

Substituting the optimal choice for d in the equation above yields

$$\frac{\beta}{\ln\left(\frac{\beta}{2\pi e}\right)} \geq \frac{2n \ln q}{\left(\ln\left(\frac{\beta}{2\pi e}\right) + 2\ln\left(\frac{q}{\sigma\sqrt{2\pi e}}\right) + \sqrt{\frac{n}{2\ln q} \cdot \frac{\ln\left(\frac{\beta}{2\pi e}\right)}{\beta}}\ln(q/\zeta)\right)^2} \tag{6}$$

Our goal is to express $\ln\left(\frac{\beta}{2\pi e}\right)$ via n, q, σ_e, σ_s and substitute the obtained expression in Equation (6). Asymptotically, assuming ζ, σ_e are constants and $\ln q \geq \ln \beta$, the above inequality is of the form $\beta/\ln(\beta) \geq \frac{d}{\ln(q)}$. Solutions for such inequality do not have closed form expressions, however, one can check that they all belong to $\Theta\left(\frac{n}{\ln(q)}\ln\left(\frac{n}{\ln q}\right)\right)$. Experiments suggest the that constant inside the Θ-notation is around 2.

Letting $X := \frac{\beta}{\ln\left(\frac{\beta}{2\pi e}\right)}$, $A := 2n\ln q$, $B = 2\ln\left(\frac{q}{\sigma\sqrt{2\pi e}}\right) + \ln\left(\frac{2n}{\ln q}\ln\left(\frac{n}{\ln q}\right)\right)$, $C := \frac{n}{2\ln q}$, $D := \ln(q/\zeta)$, Equation (6) translates to $X = \frac{A}{\left(B - 2D\sqrt{\frac{C}{X}}\right)^2}$. A positive solution to this quadratic (in $\sqrt{X}$) equation is $\sqrt{X} = \frac{2D\sqrt{C}+\sqrt{A}}{B}$. Note that the right hand side is independent of β. Unrolling the definition of X, we obtain $\frac{\beta}{\ln(\beta/(2\pi e))} = \left(\frac{2D\sqrt{C}+\sqrt{A}}{B}\right)^2$. There is no closed form solution to this equation, however, we can express the solution via the Lambert-W function[8],

[8] https://en.wikipedia.org/wiki/Lambert_W_function

which can be evaluated numerically for our parameters. Concretely, we obtain $\beta = 2\pi e^{1-W_1\left(-\frac{2\pi e B}{2D\sqrt{C}+\sqrt{A}}\right)}$, where $W_1()$ denotes the "lower" branch of Lambert-W function. It follows $\ln\left(\frac{\beta}{2\pi e}\right) = -W_1\left(-\frac{2\pi e B}{2D\sqrt{C}+\sqrt{A}}\right)$. Substituting this result in Equation (6), we obtain a closed expression for β (technically, it is a lower bound for β, but we treat it as equality):

$$\beta = \frac{2n\ln q \cdot \left(-W_1\left(-\frac{2\pi e B}{2D\sqrt{C}+\sqrt{A}}\right)\right)}{\left(-W_1\left(-\frac{2\pi e B}{2D\sqrt{C}+\sqrt{A}}\right) + 2\ln\left(\frac{q}{\sigma\sqrt{2\pi e}}\right) + \sqrt{\frac{n}{2\ln q}} \cdot \frac{B}{2D\sqrt{C}+\sqrt{A}}\ln(q/\zeta)\right)^2} \quad (7)$$

Having β (and optimal d), we obtain the expression for the security level λ achieved by the LWE parameters n, q, σ_e, σ_s:

$$\lambda = \log(T_{\text{BKZ}}(\beta, n), 2) = 0.292\beta + \log(8d, 2) + 16.4. \quad (8)$$

In the next section, we show that this formula gives very close results to the Lattice Estimator predictions, and hence we can use it to express the LWE dimension n.

Expressing n. Expressing n via $\lambda, q, \sigma_e, \sigma_s$ using Equation (5) and the approximation $\ln(\beta) \approx \ln(\lambda/0.292)$, we obtain

$$n = \frac{\lambda \cdot \left(\ln\left(\frac{\lambda}{0.584\pi e}\right) + \ln\left(\frac{q}{2\pi e \cdot \sigma_e \sigma_s}\right)\right)^2}{0.584\ln(\lambda/(0.584\pi e))\ln(q)} - \texttt{non\_leading\_order\_term}, \quad (9)$$

where `non_leading_order_term` is omitted due to its size and can be found in the scripts from our Github repository. In the next section, we provide experimental evidence of our formula's accuracy and a more compact expression for n.

3.2 The uSVP attack

Another approach to evaluate the hardness of LWE is to model the problem of finding a unique shortest vector (uSVP) in a lattice closely related to $\mathcal{L}_{\texttt{bdd}}$ [4, 6]. The uSVP attack extends $\mathcal{L}_{\texttt{bdd}}$ by embedding the vector $\mathbf{b}$ in it [37] :

$$\mathcal{L}_{\texttt{uSVP}} = \{\mathbf{v} \in \mathbb{Z}^{d+1} \mid [\mathbf{A}|\mathbf{I}_m| - \mathbf{b}]\mathbf{v} = 0 \bmod q\},$$

where as before $d = m + n$, later we optimize for d. The lattice $\mathcal{L}_{\texttt{uSVP}}$ admits the following basis matrix (written column-wise):

$$\mathbf{B}_{\texttt{uSVP}} = \begin{pmatrix} \zeta\mathbf{I}_n & \mathbf{0} & \mathbf{0} \\ \mathbf{A} & q\mathbf{I}_m & \mathbf{b} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix},$$

where again $\zeta = \max\{1, \lfloor \sigma_e/\sigma_s \rceil\}$ is the scaling constant to "balance" the $\mathbf{s}$ and $\mathbf{e}$ components of the shortest vector $(\zeta\mathbf{s}| - \mathbf{e}| - 1) \in \mathcal{L}_{\texttt{uSVP}}$. The constant 1 in the lattice is again a conventional practical choice [4].

The primal uSVP attack consist of running the BKZ lattice reduction algorithm [50, 52] on the aforementioned basis of $\mathcal{L}_{\texttt{uSVP}}$. The estimates [4, 6] predicts that BKZ succeeds in finding $(\zeta\mathbf{s}| - \mathbf{e}| - 1)$ if

$$\sqrt{\beta/(d)}\|(\zeta\mathbf{s}| - \mathbf{e}| - 1)\| \approx \sqrt{\beta}\sigma_e \leq \delta^{2\beta-(n+m+1)} \det(\mathcal{L}_{\texttt{uSVP}})^{1/d}.$$

From the shape of the basis $\mathbf{B}_{\texttt{uSVP}}$ of $\mathcal{L}_{\texttt{uSVP}}$, computing its volume (from now on we ignore the $+1$ in the dimension of $\mathcal{L}_{\texttt{uSVP}}$ and simplify it to $\dim(\mathcal{L}_{\texttt{uSVP}}) = n + m =: d$) leads to

$$\sqrt{\beta}\sigma_e \leq \delta^{2\beta-(d)} \cdot \zeta^{\frac{n}{d}} \cdot q^{1-\frac{n}{d}}. \tag{10}$$

Now let us obtain a closed form for β as a function of the LWE parameters. The following derivations are rather technical, the reader may jump directly to Equation (13) for the final result.

As in case of BDD, an attacker is allowed to choose m – the number of LWE samples to build the lattice from. As our objective is to reach the condition above for as small β as possible (the lower the β is, the easier is the attack) we aim at finding m that maximizes the right-hand size of Inequality (10). The maximum is achieved for $d = \sqrt{\frac{n\ln(q/\zeta)}{\ln\delta_\beta}}$. Substituting it in the Inequality (10) and taking logarithms leads to the success condition:

$$2\beta\ln\delta - 2\sqrt{n\ln(q/\zeta\ln\delta)} + \ln(q/\sigma_e) - \frac{1}{2}\ln\beta \geq 0.$$

Using the approximation $\ln(\delta) \approx \frac{\ln(\beta/(2\pi e))}{2\beta}$, obtain the condition on β (we keep the constants as they turn out to matter for the final result):

$$\beta \geq \frac{2n\ln(q/\zeta)\ln(\beta/(2\pi e))}{\ln^2(q\sqrt{\beta}/(2\pi e\sigma_e))}.$$

For the FHE parameters, the modulus q is chosen to be much larger than n and m and hence, larger than β. Therefore, asymptotically, the right-hand side of the inequality above belongs to $\Theta\left(\frac{n}{\ln q}\ln\left(\frac{n}{\ln q}\right)\right)$. This leads us to (again, as in BDD, the equation below is rather the inequality giving the lower bound on successfully β):

$$\beta = \frac{2n\ln(q/\zeta)\ln\left(\frac{n\ln(n/\ln q)}{2\pi e\ln(q/\sigma_e)}\right)}{\ln^2\left(\frac{q\sqrt{n\ln(n/\ln(q/\sigma_e))/\ln q}}{2\pi e\sigma_e}\right)} \tag{11}$$

Comparing this result with Equation (7), we notice that asymptotically both expressions for β in BDD and in uSVP attack match.

Substituting Equation (11), obtain the expression for λ

$$\lambda = 0.292\beta + \ln\left(8\sqrt{\frac{2n\ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}}\right) + 16.4. \tag{12}$$

Expressing n. Analogously to BDD, we express n as a function of $\lambda, q, \sigma_e, \sigma_s$:

$$n = \frac{\lambda \cdot \left(0.5\ln(\lambda/0.292) + \ln\left(\frac{q}{2\pi e \cdot \sigma_e}\right)\right)^2}{0.584\ln(\lambda/(0.584\pi e))\ln(q\sigma_s/\sigma_e)} - \texttt{non\_leading\_order\_term}, \quad (13)$$

where `non_leading_order_term` is omitted and can be found in our Github repository.

4 Fine-tuning and Verification

4.1 Our methodology

As we have detailed in the previous section, during the derivation of our formulas, several simplifications had to be made in order to express the security parameter λ via LWE parameters, and, inversely, the LWE dimension n via $\lambda, q, \sigma_e, \sigma_e$. Although our formulas perform very well 'by default', we can optimize them and compensate for the loss in accuracy coming from the simplifications via a fitting function. The idea is to add certain parameters to our formulas and then learn them by using a list of points computed from the Lattice Estimator [5] and a fitting function. We remark that the simplifications only have a noticeable effect on the non-leading terms, and they perform very well by 'default'. Thus, the correction done via the fitting function can be understood as fixing these terms.

Database. The database used to verify our formulas has been constructed as follows. Fix $\sigma_e = 3.19$. Given a range of values for q, a range for LWE dimension n and $\sigma_s \in \{\mathcal{U}_2, \mathcal{U}_3\}$, we run the Lattice Estimator to obtain the security level of the corresponding points. It is worth noticing that $\sigma_s = \mathcal{U}_2$ is employed in TFHE-like schemes where $2^{10} \leq n \leq 2^{11}$, while $\sigma_s = \mathcal{U}_3$ is utilized in the other schemes (BGV, BFV and CKKS), where the dimension n is much bigger, i.e. $n \leq 2^{16}$. We have selected various parameter sets providing different security levels to validate our formulas exhaustively. Following common practice in the FHE literature we populate our database with parameters offering at least 80 bits of security [21, 22, 43]. Table 1 shows the number of points that we considered.

σ_s	**Range of n**	**Range of $\log q$**	σ_e	**Num. points**
$\mathcal{U}_2$	$[2^{10}, 2^{11}]$	$[20, 64]$	3.19	42962
$\mathcal{U}_3$	$[2^{10}, 2^{15}]$	$[10, 1600]$	3.19	5282

Table 1: Number of points (in our database) used to verify our formulas divided by secret distribution. Half of them correspond to the output of the lattice Estimator for uSVP and the other half for BDD.

Classification and Curation Given the database, we classify the points per security level. It is important to notice that, given a security level, not all points need to be considered since most of them will never be used in practice. The considered points follow this criterion:

- Fix a LWE dimension n, we consider the point (n, q) with the biggest possible q. We can perform more computations with a bigger q.
- Fix a modulus q, we will only consider the point (n, q) with the smallest possible n. We have higher efficiency with a smaller n.

Verification. The verification step consists of plotting the curated points against our optimized formulas. Since we provide formulas derived from the attacks against uSVP and BDD, we verify each formula separately against the points where the security level corresponds to that attack.

Fine-tuning. After creating our database by running the Lattice Estimator as explained above, we do the following:

1. We refine the resulting formulas (Equations (8), (9), (12) and (13)) by incorporating additional variables. Using *coupled optimization*[9], we determine the optimal values for these variables to ensure that our parameterized functions follows the data points generated with the Lattice Estimator, i.e., accurately reflects the security level estimation.
2. Finally, we provide a further simplification of these formulas, explicitly depending on the macro variables n, λ and q. Note that in this case, the variables found using the coupled optimization technique are intrinsically dependent on the secret distribution χ_s (and so on ζ).

4.2 Verification of uSVP security level, Equation (12)

Starting from Equation (12) and using the process explained above, the resulting function for λ (considering the uSVP attack) is

$$\lambda = A\beta + B \ln\left(\frac{2n\ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}\right) + C, \tag{14}$$

where

$$\begin{array}{llll} \mathrm{A} = 0.28862 & \mathrm{B} = 1.33981 & \mathrm{C} = 5.61427 & \text{if } \chi_s = \mathcal{U}_2 \\ \mathrm{A} = 0.296208 & \mathrm{B} = 0.800603 & \mathrm{C} = 12.09086 & \text{if } \chi_s = \mathcal{U}_3. \end{array}$$

Now, our aim is to express Equation (14) in a simplified form that explicitly depends on the variables n and q.

Let define $x = n/\ln q$, $k_1 = \frac{1}{2\pi e}$ and $k_2 = \frac{1}{2\pi e \sigma_e} = \frac{k_1}{\sigma_e}$, then since $\ln(q/\zeta) \approx \ln(q/\sigma_e) \approx \ln(q)$, we have that Equation (11) can be approximate as

$$\beta \geq \frac{2n\ln(q/\zeta)\ln(k_1 x \ln x)}{\ln^2(k_2 q\sqrt{x\ln x})} \approx \frac{2n\ln q(\ln(x\ln x) - 2.8)}{(\ln q + 0.5\ln(x\ln x) - 4)^2}.$$

Considering n, q such that the security level is between 80 and 130, we have that $\ln q + 0.5\ln(x\ln x) - 4 \approx \ln q$. So

$$\beta \approx 2x\left(\ln(x) + \ln(\ln(x)) - 2.8\right). \tag{15}$$

[9] Specifically, we use the LMFIT Minimizer class: https://lmfit.github.io/lmfit-py/fitting.html.

Substituting Equation (15) in Equation (14) we have:

$$\begin{aligned}\lambda &\approx 2A\ln\left(\tfrac{n}{\ln q}+\ln\left(\ln\left(\tfrac{n}{\ln q}\right)\right)-2.8\right)\tfrac{n}{\ln q}+B\ln\left(\tfrac{2n\ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}\right)+C\\ &\approx A'\ln\left(k_3\tfrac{n}{\ln q}\right)\tfrac{n}{\ln q}+B\ln\left(\tfrac{2n\ln(q)\beta}{\ln(\beta)-2.8}\right)+C\\ &\approx A'\ln\left(k_3\tfrac{n}{\ln q}\right)\tfrac{n}{\ln q}+B\ln(4n^2k_4)+C,\end{aligned}$$

where k_3 and k_4 are small constants since if we consider n, q such that the security level is between 80 and 130,

$$k_4 = \frac{\ln x+\ln\big(\ln x\big)-2.8}{\ln\big(2x\big)+\ln\Big(\ln x+\ln\big(\ln x\big)-2.8\Big)-2.8}\approx 1.$$

Using coupled optimization, we find the following

$$\lambda \approx A\ln\left(\frac{Bn}{\ln q}\right)\frac{n}{\ln q}+C\ln n+D \tag{16}$$

$$\begin{aligned}&\text{A} = 0.445309\ \ \text{B} = 1.486982\ \ \text{C} = 0.950115\ \ \text{D} = 11.21416\ \ \text{if } \chi_s = \mathcal{U}_2\\ &\text{A} = 0.833542\ \ \text{B} = 0.154947\ \ \text{C} = 1.469823\ \ \text{D} = 18.09877\ \ \text{if } \chi_s = \mathcal{U}_3.\end{aligned}$$

The comparison results between the output of the Lattice Estimator and our formulas (Equations (14) and (16)) are presented in Tables 2 and 3, demonstrating the effectiveness of our approach in accurately estimating security levels.

$n = 2^{10}$				$n = 2^{11}$			
$\log q$	Estimator	(14)	(16)	$\log q$	Estimator	(14)	(16)
20	172	172	172	37	191	191	188
24	142	142	142	46	151	150	149
25	136	136	136	50	137	137	136
26	130	130	130	53	129	129	128
27	125	125	125	54	126	127	126
28	120	120	120	57	119	119	119
30	112	112	112	62	110	109	109
33	101	101	101	67	100	101	101
37	90	90	90	74	90	91	91
42	79	79	80	84	80	80	80

Table 2: Comparison between the security level provided by our formulas (Equations (14) and (16)) and the Lattice Estimator with with $\sigma_s = \mathcal{U}_2$.

	$n = 2^{10}$				$n = 2^{15}$		
$\log q$	Estimator	(14)	(16)	$\log q$	Estimator	(14)	(16)
16	231	230	233	650	179	180	180
18	204	203	202	760	151	151	151
19	193	191	190	810	140	141	140
25	143	141	137	880	128	129	128
27	132	129	126	930	121	121	121
28	126	124	121	1000	112	112	112
30	117	115	112	1050	106	107	106
32	109	107	104	1200	93	93	93
43	80	78	76	1400	80	80	80
48	71	70	68	1500	74	75	75

Table 3: Comparison between the security level provided by our formulas (Equations (14) and (16)) and the Lattice Estimator with $\sigma_s = \mathcal{U}_3$.

In Figure 1 we pictured the data points of the Lattice Estimator and our formula proposed in Equation (16).

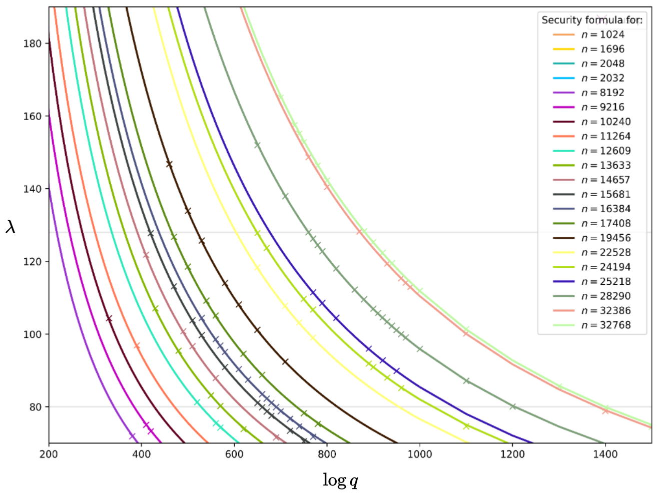

Fig. 1: The security formula (Equation (16)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$ considering the uSVP attack.

4.3 Verification of BDD security level, Equation (8)

Starting from Equation (8) and using the couple optimization, the resulting function for λ (considering the BDD attack) is

$$\lambda \approx A\beta + B \ln\left(\frac{2n\beta \ln(q/\zeta)}{\ln(\beta)}\right) + C. \tag{17}$$

where

$$\begin{array}{llll} \mathrm{A} = 0.26497 & \mathrm{B} = 3.25511 & \mathrm{C} = -13.69437 & \text{if } \chi_s = \mathcal{U}_2. \\ \mathrm{A} = 0.337349 & \mathrm{B} = 1.712118 & \mathrm{C} = 0.000003 & \text{if } \chi_s = \mathcal{U}_3 \end{array}$$

In Figures 2 and 3 we pictured the data points of the Lattice Estimator and our formula proposed in Equation (17).

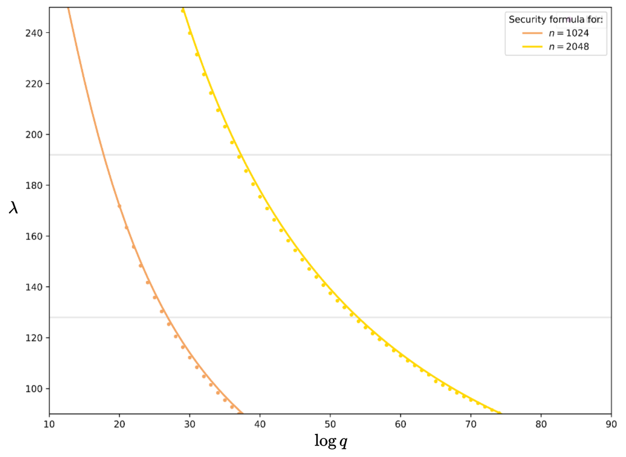

Fig. 2: The security level formula (Equation (17)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_2$ considering the BDD attack.

From the Lattice Estimator outputs considered in this paper, we observed that for binary secret, the BDD attack always outperforms uSVP, although by a non-significant amount. Indeed, as our formulas suggest, the two attacks have very close runtimes.

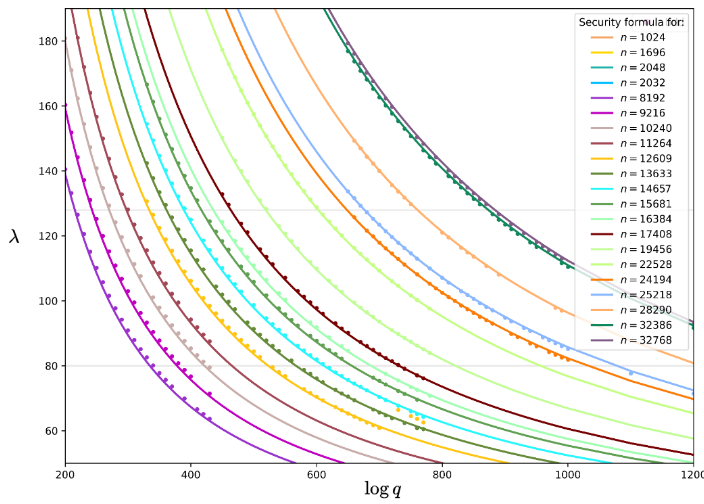

Fig. 3: The security formula (Equation (17)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$ considering the BDD attack.

4.4 Verification of the LWE dimension via uSVP, Equation (13)

Considering Equation (16) and setting $x = n/\ln q$, we have

$$\lambda \approx A\ln(Bx)\frac{n}{\ln q} + C\ln x + C\ln\ln q + D.$$

Thus,

$$n \approx \left(\frac{\lambda - C\ln x - C\ln\ln q - D}{A\ln(Bx)}\right)\ln q \approx \left(\frac{\lambda + k_1\ln\ln q}{k_2\ln(x) + k_3} + k_4\right)\ln q$$

where k_i are some constants. Since x appears only in the logarithm, we can consider the leading term of Equation (16) approximating $x \approx a\lambda + b$, where a, b are some constants. Thus, using couple optimization, we obtain

$$n \approx \left(\frac{\lambda + A\ln(\ln q)}{B\ln(\lambda) + C} + D\right)\ln q, \tag{18}$$

$\mathrm{A} = -1.142080$ $\mathrm{B} = 0.231197$ $\mathrm{C} = 1.106616$ $\mathrm{D} = -0.233138$ if $\chi_s = \mathcal{U}_2$
$\mathrm{A} = -1.073049$ $\mathrm{B} = 0.278319$ $\mathrm{C} = 0.931202$ $\mathrm{D} = 0.792882$ if $\chi_s = \mathcal{U}_3$.

The comparison results between the output of the Lattice Estimator and our formula (Equation (18)) are presented in Tables 4 and 5, demonstrating the effectiveness of our approach in accurately estimating security levels.

$\log q$	Est_λ	Est_n	(18)	$\log q$	Est_λ	Est_n	(18)
	$\lambda \approx 80$				$\lambda \approx 100$		
42	80	1024	1039	34	100	1024	1041
58	80	1408	1428	46	102	1408	1429
71	80	1728	1743	57	100	1728	1734
84	80	2048	2056	67	100	2048	2034
	$\lambda \approx 110$				$\lambda \approx 120$		
31	110	1024	1038	28	123	1024	1042
42	112	1408	1426	39	121	1408	1424
52	111	1792	1747	48	121	1792	1749
61	112	2048	2063	57	121	2048	2074
	$\lambda \approx 128$				$\lambda \approx 140$		
27	128	1024	1043	24	144	1024	1034
37	128	1408	1425	34	140	1408	1424
45	129	1728	1742	41	143	1728	1748
54	128	2048	2072	49	142	2048	2073

Table 4: Comparison between the LWE dimension provided by our formula Equation (18) and the Lattice Estimator with secret distribution $\mathcal{U}_2$. Est_n represents the n selected as input parameter for the Lattice Estimator. Est_λ represents the output security level provided by the Estimator given $\log q$, Est_n and the corresponding distribution.

$n = 2^{10} = 1024$			$n = 2^{15} = 32768$		
$\log q$	Est_λ	(18)	$\log q$	Est_λ	(18)
43	80	1063	1400	80	32801
34	102	1060	1100	101	32592
32	109	1062	1000	112	32809
29	122	1070	930	121	32908
27	132	1073	880	128	32890
25	143	1070	810	140	33012

Table 5: Comparison between the LWE dimension provided by our formula Equation (18) and the Lattice Estimator with secret distribution $\mathcal{U}_3$. Est_λ represents the output security level provided by the Estimator given $\log q$, Est_n and the corresponding distribution.

In Figure 4, we pictured the data points of the Lattice Estimator for uSVP attack and our formula proposed in Equation (18).

4.5 Verification of the LWE dimension via BDD, Equation (9)

We approximate Equation (9) using coupled optimization techniques obtaining

$$n = \left(A\frac{\lambda}{\ln \lambda} + B\ln(\ln q) + C\right)\ln q + D, \tag{19}$$

$$\begin{array}{lllll} \text{A} = 2.463040 & \text{B} = 3.426581 & \text{C} = -24.92487 & \text{D} = 128.0417 & \text{if } \chi_s = \mathcal{U}_2 \\ \text{A} = 2.368303 & \text{B} = -0.676307 & \text{C} = -4.104371 & \text{D} = -19.11047 & \text{if } \chi_s = \mathcal{U}_3. \end{array}$$

The comparison results between Equation (19) and the output of the Lattice Estimator are presented in Tables 6 and 7. In Figure 4, we show the data points of the Lattice Estimator for the uSVP attack and Equation (18).

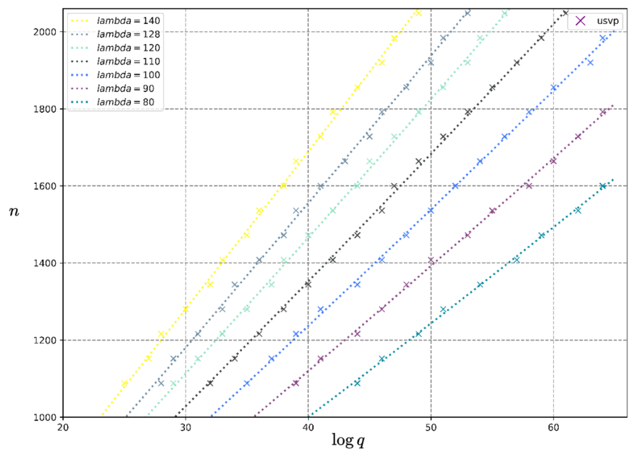

Fig. 4: Comparison between Equation (18) and the data points output by the Lattice Estimator for $\chi_s = \mathcal{U}_2$, considering the uSVP attack.

$\log q$	Est$_\lambda$	Est$_n$	(19)	$\log q$	Est$_\lambda$	Est$_n$	(19)
$\lambda \approx 80$				$\lambda \approx 100$			
42	80	1024	1039	34	100	1024	1041
58	80	1408	1428	46	102	1408	1429
71	80	1728	1742	57	100	1728	1734
84	80	2048	2056	67	100	2048	2034
$\lambda \approx 110$				$\lambda \approx 120$			
31	110	1024	1038	28	123	1024	1041
42	112	1408	1426	39	121	1408	1424
52	111	1792	1748	48	121	1792	1749
61	112	2048	2062	57	121	2048	2073
$\lambda \approx 128$				$\lambda \approx 140$			
27	128	1024	1042	24	144	1024	1034
37	128	1408	1424	34	140	1408	1424
45	129	1728	1742	41	143	1728	1748
54	128	2048	2072	49	142	2048	2072

Table 6: Comparison between the LWE dimension n provided by Equation (19) and the Lattice estimator with $\chi_s = \mathcal{U}_2$, for the BDD attack. Est$_n$ represents the n selected as the input parameter for the Lattice Estimator. Est$_\lambda$ represents the output security level provided by the Estimator given $\log q$, Est$_n$ and the corresponding distribution.

$n = 2^{10} = 1024$			$n = 2^{15} = 32768$		
$\log q$	Est_λ	(19)	$\log q$	Est_λ	(19)
43	79	1066	1450	77	33351
34	101	1055	1150	97	33136
32	108	1055	1050	106	32929
29	120	1042	930	121	33034
27	129	1043	870	129	32805
25	140	1039	810	140	32943

Table 7: Comparison between the LWE dimension n provided by our formulas (Equation (19)) and the Lattice Estimator with $\chi_s = \mathcal{U}_3$, for the BDD attack. Est_λ represents the output security level provided by the Estimator given $\log q$, Est_n and the corresponding distribution.

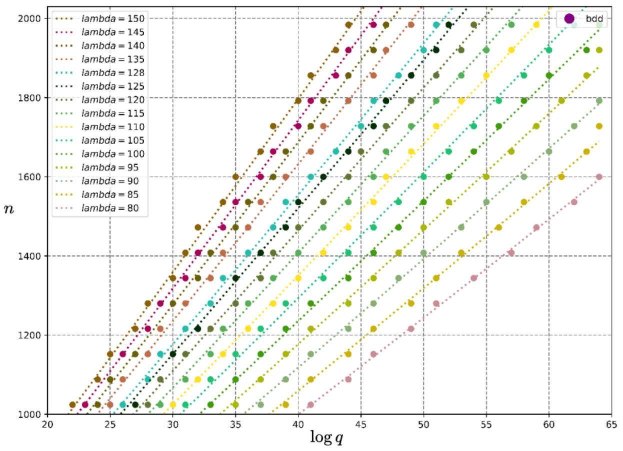

Fig. 5: The LWE dimension n (Equation (19)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$ for the BDD attack.

4.6 Comparison with [10]

In [10], the authors detail a framework to find optimal parameters for applications built from TFHE-like schemes. They find parameters which are both secure and provide correctness of computation for the underlying cryptographic task. Their method relies on a *security oracle*, which given n, q, λ and σ_s outputs the minimal σ_e that guarantees security λ. In practice[10], this oracle is constructed as a linear approximation. Their methodology is the following. Fix $\log q = 64$, $\sigma_s = \mathcal{U}_2$ and security level λ. Given a range of values for σ_e, iterate over different values of n to find the minimum n for which the Lattice Estimator outputs security level λ. The output is then a collection of points $\{(n^i, \sigma_e^i)\}_i$ which can be

[10] See https://github.com/zama-ai/concrete/tree/main/tools/parameter-curves

linearly interpolated, obtaining parameters a, b. The oracle corresponds to the function $\mathcal{F}(n) = 2^{\lceil a \cdot n + b \rceil}$. Our methodology deviates considerably from [10]. The main difference is that our formulas do not come solely from empirical results but from the mathematical descriptions of the attacks against uSVP and BDD. A more detailed comparison can be found in the extended version of the paper [38].

4.7 How to use our results in practice

We provide our formulas that relate the macro parameters $n, q, \sigma_e, \sigma_s, \lambda$ as a tool in our Github repository[11]. Our code emulates the workflow of the Lattice Estimator for the uSVP and BDD attacks. That is, given n, q, σ_s and σ_e, one can use our tool to obtain a close approximation of the security level λ, without the need to run the Estimator. Moreover, given λ, q, σ_s and σ_e our tool can be used to obtain a n equal to or close to the optimal value for the given parameter set. Once the user has greatly narrowed down their possible choices of parameters using our tool, the recommended practice is to check them against the Estimator. Notice that our tool targets *security* and does not take into account *correctness* of computation. That is, we provide a shortcut to finding secure parameters for FHE schemes at the macro level. These parameters will not guarantee the correctness of computation as this will depend on the circuit being evaluated and the chosen FHE scheme.

5 Conclusion

We provided a pioneering methodology to obtain closed formulas for the security level of LWE as a function of the LWE dimension n, modulus q, standard deviations of secret σ_s, and error σ_e. From these formulas, we can express n as a function of q, σ_s, σ_e and the security level λ. We have extensively tested and verified our formulas against empirical data obtained from the Lattice Estimator [5].

The results obtained in this work significantly accelerate the parameter selection process of any LWE-based encryption scheme. We use them to build a practical and efficient tool for researchers and practitioners deploying FHE in real-world applications. Unlike the current slow, cumbersome and rigid parameter selection process of FHE parameters, our formulas open the door to a fast, user-friendly and easily adaptable paradigm.

We are positive that the methodology detailed in this work provides a good starting point for the derivation of q and/or σ_e as functions of the remaining parameters. We leave this task for future development of our tool.

Acknowledgements

The third listed author was supported by the Spanish Ministry of Science and Innovation under Grant Agreement No RTI2018-102112-B-I00.

[11] https://github.com/sergirovira/fastparameterselection

Bibliography

[1] Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys (CSUR) **51**(4), 1–35 (2018)

[2] Albrecht, M.R.: On Dual Lattice Attacks Against Small-Secret LWE and Parameter Choices in HElib and SEAL. In: Advances in Cryptology – EUROCRYPT 2017. pp. 103–129 (2017)

[3] Albrecht, M.R., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., Lokam, S., Micciancio, D., Moody, D., Morrison, T., Sahai, A., Vaikuntanathan, V.: Homomorphic encryption security standard. Tech. rep., HomomorphicEncryption.org, Toronto, Canada (November 2018)

[4] Albrecht, M.R., Göpfert, F., Virdia, F., Wunderer, T.: Revisiting the expected cost of solving uSVP and applications to LWE. In: Advances in Cryptology–ASIACRYPT 2017. pp. 297–322 (2017)

[5] Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015)

[6] Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-Quantum Key Exchange: A New Hope. In: Proceedings of the 25th USENIX Conference on Security Symposium. p. 327–343 (2016)

[7] Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica **6**(1), 1–13 (1986)

[8] Badawi, A.A., Bates, J., Bergamaschi, F., Cousins, D.B., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., Liu, Z., Micciancio, D., Quah, I., Polyakov, Y., R.V., S., Rohloff, K., Saylor, J., Suponitsky, D., Triplett, M., Vaikuntanathan, V., Zucca, V.: OpenFHE: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915 (2022)

[9] Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: SODA 2016. pp. 10–24. SIAM (2016)

[10] Bergerat, L., Boudi, A., Bourgerie, Q., Chillotti, I., Ligier, D., Orfila, J.B., Tap, S.: Parameter Optimization and Larger Precision for (T)FHE. Journal of Cryptology **36**(3), 28 (2023)

[11] Biasioli, B., Marcolla, C., Calderini, M., Mono, J.: Improving and Automating BFV Parameters Selection: An Average-Case Approach. Cryptology ePrint Archive, Paper 2023/600 (2023)

[12] Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In: 2018 IEEE EuroS&P. pp. 353–367 (2018)

[13] Brakerski, Z.: Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In: Advances in Cryptology – CRYPTO 2012. pp. 868–886 (2012)
[14] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
[15] Carpov, S., Dubrulle, P., Sirdey, R.: Armadillo: a compilation chain for privacy preserving applications. In: Proceedings of the 3rd International Workshop on Security in Cloud Computing. pp. 13–19 (2015)
[16] Cheon, J.H., Costache, A., Moreno, R.C., Dai, W., Gama, N., Georgieva, M., Halevi, S., Kim, M., Kim, S., Laine, K., Polyakov, Y., Song, Y.: Introduction to homomorphic encryption and schemes. Protecting Privacy through Homomorphic Encryption pp. 3–28 (2021)
[17] Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: A full RNS variant of approximate homomorphic encryption. In: International Conference on Selected Areas in Cryptography – SAC 2018. pp. 347–368. Springer (2018)
[18] Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic Encryption for Arithmetic of Approximate Numbers. In: Advances in Cryptology – ASIACRYPT 2017. pp. 409–437 (2017)
[19] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: international conference on the theory and application of cryptology and information security. pp. 3–33. Springer (2016)
[20] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast Fully Homomorphic Encryption over the Torus. Journal of Cryptology **33**(1), 34–91 (2020)
[21] Costache, A., Smart, N.P.: Which ring based somewhat homomorphic encryption scheme is best? In: Cryptographers' Track at the RSA Conference. pp. 325–340. Springer (2016)
[22] Costache, A., Smart, N.P.: Homomorphic encryption without gaussian noise. Cryptology ePrint Archive, Paper 2017/163 (2017)
[23] Crockett, E., Peikert, C., Sharp, C.: Alchemy: A language and compiler for homomorphic encryption made easy. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 1020–1037 (2018)
[24] Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: Lwe with side information: Attacks and concrete security estimation. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. pp. 329–358. Springer International Publishing, Cham (2020)
[25] Dathathri, R., Kostova, B., Saarikivi, O., Dai, W., Laine, K., Musuvathi, M.: EVA: An encrypted vector arithmetic language and compiler for efficient homomorphic computation. In: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 546–561 (2020)
[26] Ducas, L., Pulles, L.N.: Accurate score prediction for dual-sieve attacks. Cryptology ePrint Archive, Report 2023/1850 (2023)

[27] Ducas, L., Pulles, L.N.: Does the dual-sieve attack on learning with errors even work? In: Annual International Cryptology Conference. pp. 37–69. Springer (2023)
[28] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(1), 238–268 (Feb 2018)
[29] van Elsloo, T., Patrini, G., Ivey-Law, H.: SEALion: A framework for neural network inference on encrypted data. arXiv preprint arXiv:1904.12840 (2019)
[30] Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive (2012)
[31] Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhan, Z.: FALCON: Fast-fourier lattice-based compact signatures over NTRU. Submission to the NIST's post-quantum cryptography standardization process **36**(5), 1–75 (2018)
[32] Gentry, C.: A fully homomorphic encryption scheme, vol. 20. Stanford university Stanford (2009)
[33] Guo, Q., Johansson, T.: Faster dual lattice attacks for solving LWE with applications to CRYSTALS. In: Advances in Cryptology–ASIACRYPT 2021. pp. 33–62. Springer (2021)
[34] Hanrot, G., Pujol, X., Stehlé, D.: Analyzing blockwise lattice algorithms using dynamical systems. In: Advances in Cryptology – CRYPTO 2011. pp. 447–464 (2011)
[35] Herold, G., Kirshanova, E., May, A.: On the asymptotic complexity of solving lwe. Des. Codes Cryptography **86**(1), 55–83 (jan 2018)
[36] Howgrave-Graham, N.: A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. In: Advances in Cryptology - CRYPTO 2007. pp. 150–169 (2007)
[37] Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proceedings of the fifteenth annual ACM symposium on Theory of computing. pp. 193–206 (1983)
[38] Kirshanova, E., Marcolla, C., Rovira, S.: Guidance for efficient selection of secure parameters for fully homomorphic encryption (2024), to appear
[39] Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Des. Codes Cryptography **75**(3), 565–599 (jun 2015)
[40] Lattigo: http://github.com/ldsec/lattigo
[41] Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: Cryptographers' Track at the RSA Conference. pp. 293–309. Springer (2013)
[42] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 1–23 (2010)
[43] Ma, S., Huang, T., Wang, A., Wang, X.: Accelerating BGV bootstrapping for large p using null polynomials over $\mathbb{Z}_{p^e}$. Cryptology ePrint Archive, Paper 2024/115 (2024)

[44] Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., Fitzek, F.H., Aaraj, N.: Survey on fully homomorphic encryption, theory, and applications. Proceedings of the IEEE **110**(10), 1572–1609 (2022)

[45] Martins, P., Sousa, L., Mariano, A.: A survey on fully homomorphic encryption: An engineering perspective. ACM Computing Surveys (CSUR) **50**(6), 1–33 (2017)

[46] MATZOV: Report on the security of LWE: Improved dual lattice attack (April 2022), https://zenodo.org/records/6412487

[47] Mono, J., Marcolla, C., Land, G., Güneysu, T., Aaraj, N.: Finding and evaluating parameters for bgv. In: International Conference on Cryptology in Africa – AFRICACRYPT 2023. pp. 370–394. Springer (2023)

[48] Paillier, P.: Invited talk: Recent advances in homomorphic compilation, https://youtu.be/phWYLwlPTY0?si=gwcf8svL6tOYcizv

[49] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing. pp. 84–93 (2005)

[50] Schnorr, C.P.: A hierarchy of polynomial time lattice basis reduction algorithms. Theoretical Computer Science **53**(2), 201–224 (1987)

[51] Schnorr, C.P.: Lattice reduction by random sampling and birthday methods. In: Alt, H., Habib, M. (eds.) STACS 2003. pp. 145–156 (2003)

[52] Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical programming **66**(1-3), 181–199 (1994)

[53] Microsoft SEAL (release 3.4). https://github.com/Microsoft/SEAL (Oct 2019), microsoft Research, Redmond, WA.

[54] Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 617–635. Springer (2009)

[55] Viand, A., Jattke, P., Hithnawi, A.: SoK: Fully Homomorphic Encryption Compilers. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1092–1108. IEEE Computer Society (2021)