A Tool for Fast and Secure LWE Parameter Selection: the FHE case

Beatrice Biasioli[1], 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 in related fields, such as machine learning, are increasingly interested in using FHE to provide privacy to their applications.
Despite this progress, selecting secure and efficient parameters for FHE remains a complex and challenging task due to the intricate interdependencies between parameters. In this work, we address this issue by providing a rigorous theoretical foundation for parameter selection for any LWE-based schemes, with a specific focus on FHE. Our approach starts with an in-depth analysis of lattice attacks on the LWE problem, deriving precise expressions for the most effective ones. Building on this, we introduce closed-form formulas that establish the relationships among the LWE parameters.
In addition, we introduce a numerical method to enable the accurate selection of any configurable parameter to meet a desired security level. Finally, we use our results to build a practical and efficient tool for researchers and practitioners deploying FHE in real-world applications, ensuring that our approach is both rigorous and accessible.

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 [50].

The first FHE scheme was introduced in 2009 by Gentry [37]. 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, such as BGV [15], BFV [14, 35], FHEW [28], TFHE [19, 20], and CKKS [17, 18]. More details on FHE and its applications can be found in surveys [1, 50, 51].

The security of all practical FHE schemes is based on the presumed intractability of the (decision) Learning with Errors (LWE) problem, [56], and its ring variant (RLWE) [48]. Informally, the decisional version of LWE consists of 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].

The problem arising from lattice-based constructions is that the error grows whenever a homomorphic operation is performed. In particular, it grows exponentially when homomorphic multiplications are computed. However, in order to guarantee correct decryption, the error has to be small. Specifically, its maximal coefficient must be smaller than a quantity depending on the modulus q. One approach to accommodating more operations is increasing the 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.

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, the Homomorphic Encryption Standard [3] (using the Lattice Estimator[4]) provides upper bounds on the size of the modulus q for given security levels λ and dimensions n through lookup tables, recently updated in [13]. Moreover, in [53], Mono *et al.* proposed a compact formula that computes the hardness of LWE for given dimension n, modulus q and the standard deviation of secret distribution σ_s. Finally, the authors of [44], starting from a theoretical analysis of lattice attacks, present closed and precise formulas for two key tasks: 1) deriving the security parameter λ given the secret distribution χ_e, n, and the modulus q, 2) determining n as a function of λ, q, and χ_e. Our results are built on this work, which we extend and improve in several directions.

In addition to these general efforts, researchers have also focused on optimizing parameters for *specific* FHE schemes. 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 according to some predefined requirements [50, 62]. Some examples are ALCHEMY [23], Cingulata

[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.

[4] The Lattice Estimator (`https://github.com/malb/lattice-estimator` [5]) 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.

[16], EVA [26] and SEALion [34]. Additionally, Bergerat *et al.* [10] proposed a framework for efficiently selecting parameters in TFHE-like schemes. In [53], the authors developed an interactive parameter generator for the leveled BGV scheme, which supports arbitrary circuit models and Biasioli *et al.* [11] further extended this approach to the BFV scheme.

While these contributions mark significant progress toward the accessible and general adoption of FHE, a fully user-friendly and efficient tool for secure parameter tuning remains unavailable. As highlighted in Paillier's invited talk [54], the field still faces the challenge of simplifying parameter selection to a point where non-cryptographic-experts can confidently implement FHE in diverse applications.

Our contribution. Building on our previous work [44], we extend this result in several directions: we additionally express the LWE parameters q and χ_e via the remaining LWE parameters and a given security level λ (here, as in the previous work, we fix the distribution of the secret to either binary or ternary). Furthermore, for even more accurate estimates, we employ numerical solvers that allow us to find precise solutions fast. All of these and more minor improvements enhance our tool, which implements the functionality of outputting LWE parameters for a given security level.

Our analysis focuses on two types of lattice algorithms: the so-called Bounded Distance Decoding (BDD) attack and the unique Shortest Vector Problem (uSVP) attack. We chose these two as they are currently the most efficient attacks, especially in the context of FHE. Certain versions of dual attacks [52] outperform the attacks we consider here for some relevant parameters, however, at the time of writing, these dual attacks do not offer correctness [30].[5]

From this rigorous theoretical analysis, we derive precise formulas that reveal the relationships among FHE macro parameters, offering faster and versatile parameter selection. Specifically,

1. For each considered attack, we
   - derive the security parameter λ as a function of the standard deviations σ_s and σ_e, the dimension n, and the modulus q.
   - express the LWE dimension n in terms of λ, q, σ_e, and σ_s,
   - express the LWE modulus q in terms of λ, n, σ_e, and σ_s,
   - express the standard deviation of the LWE noise σ_e in terms of λ, q, n, and σ_s.
2. Taking the obtained formulas as a starting point, we build more precise estimates by conducting extensive experiments with the Lattice Estimator [5], creating a large dataset that correlates n, q, σ, and λ and producing a fitting

[5] Even if [52] was correct, the improvement over uSVP or BDD would be rather marginal as can be seen by running the Lattice Estimator. The versions of dual attacks [55] that come with correctness guarantees are inferior to the attacks considered here for concrete parameters.

function that relates the LWE parameters with various security levels. This effort enables us to adjust the lower-order terms in the derived expressions, ensuring accurate estimates for broad parameter sets.

3. An alternative road towards precise estimates is numerical solvers. Since our formulas are derived from rather elaborate complexity estimates of lattice attacks, the LWE parameters are entwined, and often it is hard to derive a nice analytical solution for a specific variable. Numerical solvers, however, perform very well at this task. Employing Python's scipy `fsolve` functionality, we are able to 'reverse' the lattice estimator for any desired LWE parameters with striking precision.
4. Most importantly for practitioners, we provide a practical tool implementing these formulas and offering best-practice guidelines for their application. Written in Python and publicly available on Github repository[6], our tool ensures that our approach is rigorous, accessible, and fast.
5. We augment our tool with the option of checking for NTRU parameters to ensure that they do not lie in the insecure regime [32].

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

In [13], the authors provide tables listing parameters for FHE applications targeting different levels of security. Their work is particularly valuable to non-experts since it allows them to select secure parameters for their applications quickly. The main difference between our work and [13] is the scope of parameters that an end-user can obtain. That is, a table-based approach such as the one provided by [13] is rigid by design. Although the authors offer a way to update the parameters via a script, they are restricted to a predefined set of values. On the other hand, with our tool, we can quickly get parameters for any range that an application might require, without having to run any LWE estimator.

Advantages of a formula-based approach. 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 brute-force, which can mean

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

checking many cases until the desired 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, 46, 60]. 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 or other LWE-based primitive in real-world applications.

The structure of the paper is as follows: Section 2 introduces the notation and mathematical background necessary for understanding the paper. In Section 3, we provide a comprehensive analysis of the BDD and uSVP attacks, deriving formulas that establish the relationships among the macro parameters of FHE. These formulas are fine-tuned in Section 4, while Section 5 examines their solutions using a numerical method. Section 6 offers practical guidance on how to use our implementation, and Section 7 compares our approach with prior works. Finally, Section 8 presents our conclusions.

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 $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_k)$ 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[7] 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, [57]). Together with a lattice basis, it receives as input an

[7] 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 [39] 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

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

which is the choice adopted by [12, 33, 36]. The correcting constant of 16.4 obtained experimentally [9]. The concrete choice of $T_{\text{BKZ}}(\beta, n)$ is called the core-SVP model [6]. Our results are easy to adapt to other existing choices of $T_{\text{BKZ}}(\beta, n)$.

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), [58]). *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 [40].[8]

Lemma 1 ([40, Lemma 4]). *Let the sequence $\|\mathbf{b}_1^\star\|, \ldots, \|\mathbf{b}_n^\star\|$ 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 [56]. 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

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

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 [48, 61] and Module-LWE [45]. 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 [30] of failing heuristics

[8] Even though in [40, Lemma 4] the authors talk about *continuous* Gaussian, the result holds for the discrete Gaussian too.

employed in efficient dual attacks [38, 52] invalidate the claimed complexities. Despite of ongoing attempts to bring dual attacks back into play [29], 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, 42]. 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 [47], 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 [47] 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 $0 \le \eta < d$ 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 $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1, \dots, \mathbf{b}_{d-\eta+1})$ is shorter than the shortest vector in the projected lattice $\mathbf{B}'_{\mathtt{bdd},[d-\eta+1,d]}$ given by the basis $(\pi_{d-\eta+1}(\mathbf{b}'_{d-\eta+1}), \dots, \pi_{d-\eta+1}(\mathbf{b}'_d))$. 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, $\|\mathbf{b}_1^\star\|, \dots \|\mathbf{b}_{d-\eta+1}^\star\|$ satisfy

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

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 $T_{\text{BKZ}}(\beta, d) = 2^{0.292\beta} \cdot 8d$, while the runtime of SVP on η-dimensional lattice is $T_{\text{SVP}}(\eta) = 2^{0.292\eta}$. The two runtimes differ only by a polynomial 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

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

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

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

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 that the constant inside the Θ-notation is 1.

Letting $X := \frac{\beta}{\ln\left(\frac{\beta}{2\pi e}\right)}$, $A := 2n\ln q$, $B = 2\ln\left(\frac{q}{\sigma_e\sqrt{2\pi e}}\right) + \ln\left(\frac{2n}{\ln q}\ln\left(\frac{n}{\ln q}\right)\right)$ (it is the second addend of B where we used the simplification $\ln\left(\frac{\beta}{2\pi e}\right) \approx \ln\left(\frac{2n}{\ln q}\ln\left(\frac{n}{\ln q}\right)\right)$); $C := \frac{n}{2\ln q}$, $D := \ln(q/\zeta)$, 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[9], 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):

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

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:

$$\lambda = \log(T_{\mathrm{BKZ}}(\beta, d), 2) = 0.292\beta + \log_2(8d) + 16.4. \tag{8}$$

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. In order to express n via $\lambda, q, \sigma_e, \sigma_s$ we look at Equation (5). This is a quadratic inequality (treated as equality) in n. Out of the two roots we choose the one that gives us the matching answers for concrete choices of $n, \lambda, q, \sigma_e, \sigma_s$. The solution is of the form (recall that $\zeta = \max\{1, \lfloor\sigma_e/\sigma_s\rceil\}$)

$$n = \frac{2\ln q\cdot\beta\cdot(2\ln q + \ln(\beta/(2\pi e) - 2\ln(\sigma_e\sqrt{2\pi e}))^2}{\ln(\beta/(2\pi e))(4\ln q - 2\ln\zeta)^2}.$$

Substituting the first order approximation $\beta \approx (\lambda - \log(8d))/0.292$ as (see Equation (3)), yields

$$n = \frac{2\ln q\cdot(\lambda-\log(8d))\cdot(2\ln q + \ln((\lambda-\log(8d))/(0.584\pi e)) - 2\ln(\sigma_e\sqrt{2\pi e}))^2}{0.292\ln((\lambda-\log(8d))/(0.584\pi e))(4\ln q - 2\ln\zeta)^2}. \tag{9}$$

Expressing $\ln q$. Inspecting Equation (5), we notice that it is linear in $\ln q$ inequality (treated here as equality). Concretely,

$$\ln q = \frac{(d/\beta - 1)\ln(\beta/(2\pi e)) + 2\ln(\sigma_2\sqrt{2\pi e})}{2(1-n/d)}.$$

Substituting the approximation for $\beta \approx (\lambda - \log(8d))/0.292$ and the optimal choice for dimension d, we express $\ln\sigma_e$ as a function of $\lambda, n, \sigma_e, \sigma_s$.

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

Expressing $\ln \sigma_e$. Similarly to $\ln q$, a closer look at Equation (5) tells that this inequality (again treated as equality) is linear $\ln \sigma_e$. Concretely, we can express the exact expressions for $\ln \sigma_e$ are

$$\ln \sigma_e = \frac{\beta + 2\beta/\ln(\beta/(2\pi e))\left(\ln(q/\sqrt{2\pi e}) - \frac{n}{d}\ln q\right) - d}{2\beta/\ln(\beta/(2\pi e)},$$

or

$$\ln \sigma_e = \frac{\beta + 2\beta/\ln(\beta/(2\pi e))\left(\ln(q/\sqrt{2\pi e}) - \frac{n}{d}(\ln q + \ln \sigma_s\right) - d}{2\beta/\ln(\beta/(2\pi e)},$$

depending on whether $\zeta = 1$ (in the first case) or $\zeta = \sigma_e/\sigma_s$ (in the second case). Substituting the approximation for $\beta \approx (\lambda - \log(8d))/0.292$ and the optimal choice for dimension d, we express $\ln \sigma_e$ as a function of λ, q, n, σ_s. As the resulting expression is fairly cumbersome to write down, we omit it here. The precise expression can be found in our scripts.

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 [43] :

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

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 [57, 59] 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 (14) 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 matter for the accuracy of the final result):

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

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):

$$\beta = \frac{2n \ln(q/\zeta) \ln\left(\frac{n \ln(n/\ln q)}{2\pi e \ln(q/\sigma_e)}\right)}{\ln^2\left(\frac{q\sqrt{n \ln(n/\ln(q/\sigma_e))/\ln q}}{2\pi e \sigma_e}\right)} \tag{12}$$

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

Substituting Equation (12), obtain the expression for λ

$$\lambda = 0.292\beta + \log_2\left(8\sqrt{\frac{2n \ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}}\right) + 16.4. \tag{13}$$

Expressing n. Given the desired security level λ, for fixed q, σ_e, σ_s, we can derive the smallest n that reaches the given λ. Noticing that Ineq. (11) is linear in n, we express

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

Treating the above inequality as equality and using $\beta \approx \lambda/0.292$ as a first order approximation (see Equation (3)), yields

$$n = \frac{\lambda\left(0.5 \ln(\lambda/0.292) + \ln(q/(2\pi e \sigma_e))\right)^2}{0.584 \ln(q/\zeta) \ln(\lambda/(0.584\pi e))}. \tag{14}$$

We defer from refinements of the expression as they involve tedious computations coming form a more accurate expression of β, Equation (3). In the latter section we show that Equation (14) already provides a very good accuracy.

Expressing $\ln q$. We notice that Inequality (11) is quadratic in $\ln q$. Treating this expression as equality and choosing the positive root (which can be checked with some known parameters) reveals the simplified solution:

$$\ln q = \frac{n - \frac{\beta}{\ln(2\pi e)} \ln\left(\frac{\sqrt{\beta}}{2\pi e \sigma_e}\right) + \sqrt{n^2 - \frac{2n\beta}{\ln(\beta/(2\pi e))} \ln\left(\frac{2\pi e \sigma_e^2}{\sqrt{\beta}}\right)}}{\ln(\beta/(2\pi e)}.$$

Substituting the approximation for $\beta = \lambda/0.292$, we obtain the expression for $\ln q$ as a function of LWE parameters and λ.

Expressing $\ln \sigma_e$. Similarly, Inequality (11) is quadratic in $\ln \sigma_e$. In case $\zeta = \sigma_e/\sigma_s$, the relevant for us solution is

$$\ln \sigma_e = \frac{\frac{\beta}{\ln(\beta/(2\pi e))} \ln\left(\frac{q\sqrt{\beta}}{2\pi e}\right) - n + \sqrt{n^2 - 2n\frac{\beta}{\ln(\beta/(2\pi e))} \ln\left(\frac{\sqrt{\beta}}{2\sigma_s \pi e}\right)}}{\beta/\ln(\beta/(2\pi e))}.$$

In case $\zeta = 1$, we have

$$\ln \sigma_e = \ln\left(\frac{q\sqrt{\beta}}{2\pi e}\right) - \frac{\sqrt{2n\ln q}}{\sqrt{\beta/\ln(\beta/(2\pi e))}}.$$

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 $\lambda, q, \sigma_s, \sigma_e$. 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, 49]. 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), (13) and (14)) by incorporating additional variables. Using *coupled optimization*[10], 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 (13)

Starting from Equation (13) and using the process explained above, the resulting function for λ (considering the uSVP attack) is

$$\lambda = A\beta + B \ln\left(\frac{2n\ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}\right) + C, \tag{15}$$

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

where

$$\begin{array}{llll} A = 0.317747 & B = 2.071129 & C = 1.849214 & \text{if } \chi_s = \mathcal{U}_2 \\ A = 0.296208 & B = 0.800603 & C = 12.09086 & \text{if } \chi_s = \mathcal{U}_3. \end{array}$$

Now, our aim is to express Equation (15) 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 (12) 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

$$\beta \approx 2x\left(\ln(x) + \ln(\ln(x)) - 2.8\right). \tag{16}$$

Substituting Equation (16) in Equation (15) 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,

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

Using coupled optimization, we find the following

$$\lambda \approx \tilde{A}\ln\left(\frac{\tilde{B}n}{\ln q}\right)\frac{n}{\ln q} + \tilde{C}\ln n + \tilde{D} \tag{17}$$

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

The comparison results between the output of the Lattice Estimator and our formulas (Equations (15) and (17)) 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	(15)	(17)	$\log q$	Estimator	(15)	(17)
20	172	178	172	37	191	193	188
24	142	145	142	46	151	152	149
25	136	139	136	50	137	139	136
26	130	133	130	53	129	130	128
27	125	128	125	54	126	128	126
28	120	123	120	57	119	121	119
30	112	114	112	62	110	111	109
33	101	103	101	67	100	102	101
37	90	92	90	74	90	93	91
42	79	81	80	84	80	82	80

Table 2: Comparison between the security level provided by our formulas (Equations (15) and (17)) and the Lattice Estimator with with $\sigma_s = \mathcal{U}_2$.

$n = 2^{10}$				$n = 2^{15}$			
$\log q$	Estimator	(15)	(17)	$\log q$	Estimator	(15)	(17)
16	231	215	233	650	179	155	180
18	204	187	202	760	151	130	151
19	193	175	190	810	140	121	141
25	143	126	137	880	128	110	128
27	132	115	126	930	121	104	121
28	126	110	121	1000	112	96	112
30	117	102	112	1050	106	107	106
32	109	94	104	1200	93	80	93
43	80	68	76	1400	80	69	80
48	71	60	68	1500	74	64	75

Table 3: Comparison between the security level provided by our formulas (Equations (15) and (17)) 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 (17).

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

$$\lambda \approx \tilde{A}\beta + \tilde{B} \ln\left(\frac{2n\beta \ln(q/\zeta)}{\ln(\beta)}\right) + \tilde{C}. \tag{18}$$

where

$$\begin{aligned} &\tilde{A} = 0.26497 \quad \tilde{B} = 3.25511 \quad \tilde{C} = -13.69437 \quad \text{if } \chi_s = \mathcal{U}_2. \\ &\tilde{A} = 0.28891 \quad \tilde{B} = 0.87868 \quad \tilde{C} = 19.1069 \qquad \text{if } \chi_s = \mathcal{U}_3 \end{aligned}$$

In Figures 2 and 3 we pictured the data points of the Lattice Estimator and our formula proposed in Equation (18).

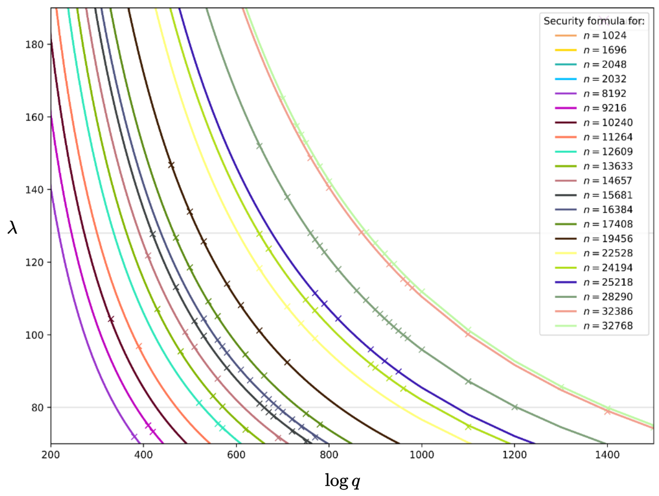


Fig. 1: The security formula (Equation (17)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$ considering the uSVP attack.

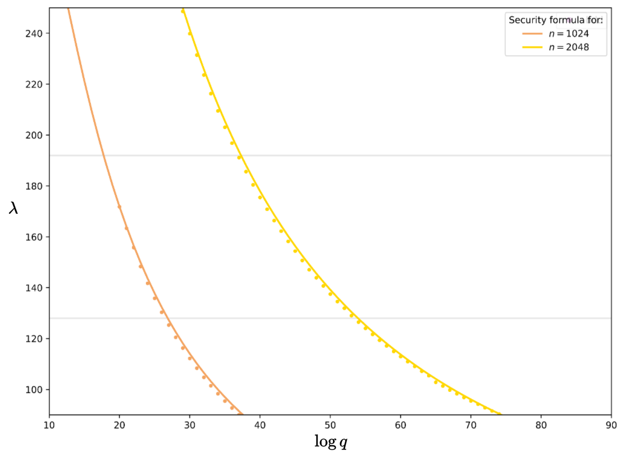


Fig. 2: The security level formula (Equation (18)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_2$ considering the BDD attack.

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.

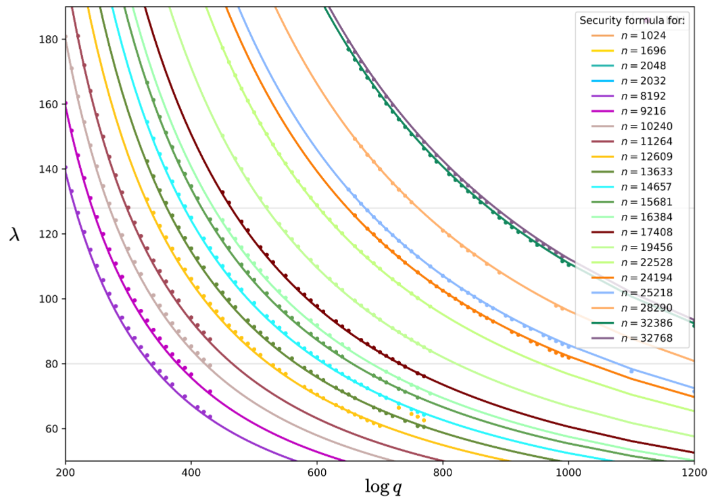


Fig. 3: The security formula (Equation (18)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$ considering the BDD attack.

Our goal is to express Equation (18) in a simplified form that explicitly depends on the variables n and q.

Since $\ln(q/\zeta) \approx \ln(q/\sigma_e) \approx \ln(q)$, we have that, starting from Equation (7),

- $A = 2n \ln q \approx n \ln q$;
- $B = 2 \ln \left(\frac{q}{\sigma\sqrt{2\pi e}}\right) + \ln \left(\frac{2n}{\ln q} \ln \left(\frac{n}{\ln q}\right)\right) \approx \ln q + \ln \left(\frac{n}{\ln q} \ln \left(\frac{n}{\ln q}\right)\right)$
- $C = \frac{n}{2 \ln q} \approx \frac{n}{\ln q}$,
- $D = \ln(q/\zeta) \approx \ln q$

Thus,

$$\frac{B}{2D\sqrt{C} + \sqrt{A}} \approx k_1 \sqrt{\frac{\ln q}{n}} + k_2$$

where k_1 and k_2 are small constants. Let $z = -2\pi e \left(k_1 \sqrt{\frac{\ln q}{n}} + k_2\right)$, then Equation (7) can be approximate as

$$\beta \approx \frac{2n \ln q \cdot (-W_1(z))}{\left(-W_1(z) + \ln q + \sqrt{\frac{n}{\ln q}} \cdot \left(k_1 \sqrt{\frac{\ln q}{n}} + k_2\right) \ln q + k_3\right)^2}$$

$$\approx \frac{2n \ln q \cdot (-W_1(z))}{\left(-W_1(z) + k_4 \ln q + k_2\sqrt{n \ln q} + k_3\right)^2}$$
$$\approx \frac{n \cdot (-W_1(z))}{\left(-\frac{1}{\sqrt{\ln q}} W_1(z) + k_4\sqrt{\ln q} + k_2\sqrt{n}\right)^2}$$

Since $W_1()$ denotes the "lower" branch of Lambert-W function and the Lambert-W is the inverse function of $y = xe^x$.

For our value of z we can *somehow* approximate $-W(z) = -\ln(z) + k$, for some constant k [41]. Thus, since $-\frac{1}{\sqrt{\ln q}} W_1(z)$ is a small constant and since $\ln\left(\frac{n}{\ln q}\right) \frac{\ln q}{n} \approx 0$, Equation (7) becomes

$$\beta \approx \frac{k_1 n \cdot \left(\ln\left(\frac{n}{\ln q}\right) + k\right)}{\left(k_4\sqrt{\ln q} + k_2\sqrt{n}\right)^2} \approx \frac{kn \cdot \left(\ln\left(\frac{n}{\ln q}\right) + k\right)}{\tilde{k}_4 \ln q + \tilde{k}_2 n + \tilde{k}_3\sqrt{n \ln q}}$$
$$\approx k_5 \frac{n}{\ln q} \ln\left(\frac{n}{\ln q} + k\right) + k_6, \tag{19}$$

where k_i are some constants.

Finally, substituting Equation (19) in Equation (18), we have

$$\lambda \approx a \frac{n}{\ln q} \ln\left(\frac{n}{\ln q} + k\right) + B \ln\left(2n^2 + d\right) + c, \tag{20}$$

Note that Equation (20) is similar to Equation (17), and this is not surprising as the two attacks yield very similar results. Therefore, we aim to further approximate Equation (20) to obtain a formula identical to Equation (17), but with different constants. Thus, using coupled optimization, we obtain

$$\lambda \approx A' \ln\left(\frac{B'n}{\ln q}\right) \frac{n}{\ln q} + C' \ln n + D' \tag{21}$$

where

$$A' = 0.424578 \quad B' = 2.122152 \quad C' = 1.959558 \quad D' = 1.155390 \quad \text{if } \chi_s = \mathcal{U}_2$$
$$A' = 0.606897 \quad B' = 0.476667 \quad C' = 0.667667 \quad D' = 15.20932 \quad \text{if } \chi_s = \mathcal{U}_3.$$

The comparison results between the output of the Lattice Estimator and our formulas (Equations (18) and (21)) are presented in Tables 4 and 5.

$n = 2^{10}$				$n = 2^{11}$			
$\log q$	Estimator	(18)	(21)	$\log q$	Estimator	(18)	(21)
20	172	166	173	37	191	185	190
24	142	138	142	46	151	147	150
25	136	132	136	50	137	135	137
26	130	127	130	53	129	127	129
27	125	122	125	54	126	125	127
28	120	117	120	57	119	118	120
30	112	109	112	62	110	108	110
33	101	99	101	67	100	100	101
37	90	88	90	74	90	91	91
42	79	77	79	84	80	80	80

Table 4: Comparison between the security level provided by our formulas (Equations (18) and (21)) and the Lattice Estimator with with $\sigma_s = \mathcal{U}_2$.

$n = 2^{10}$				$n = 2^{15}$			
$\log q$	Estimator	(18)	(21)	$\log q$	Estimator	(18)	(21)
16	231	234	232	650	179	179	179
18	204	206	202	760	151	151	150
19	193	194	190	810	140	140	140
25	143	144	140	880	128	128	128
27	132	132	128	930	121	121	120
28	126	127	123	1000	112	112	112
30	117	117	114	1050	106	107	106
32	109	109	106	1200	93	93	92
43	80	79	78	1400	80	80	79
48	71	70	70	1500	74	74	74

Table 5: Comparison between the security level provided by our formulas (Equations (18) and (21)) and the Lattice Estimator with $\sigma_s = \mathcal{U}_3$.

4.4 Verification of the LWE dimension via uSVP, Equation (14)

Starting from Equation (14) and using the couple optimization, the resulting function for n (considering the uSVP attack) is

$$n = \frac{A\lambda\big(0.5\ln(\lambda/0.292) + \ln(q/(2\pi e\sigma_e)) + B\big)^2}{0.584\ln(q/\zeta)\ln(\lambda/(0.584\pi e) + C)}, \tag{22}$$

where

$$\begin{array}{llll} A = 1.02575 & B = 0.17241 & C = 34.84910 & \text{if } \chi_s = \mathcal{U}_2 \\ A = 1.05153 & B = 0.52652 & C = 43.20997 & \text{if } \chi_s = \mathcal{U}_3. \end{array}$$

In Figure 4 we pictured the data points of the Lattice Estimator and our formula proposed in Equation (22) for the ternary distribution.

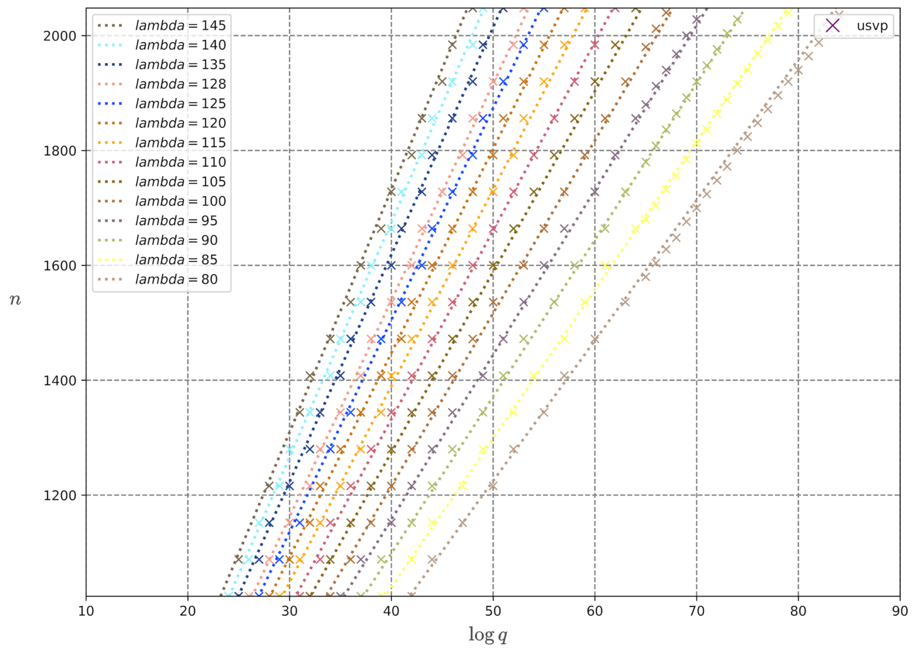


Fig. 4: The security level formula (Equation (22)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$ considering the uSVP attack.

Our goal is to express Equation (22) in a simplified form that explicitly depends on the variables λ and q. To do this, we consider Equation (17) and setting $x = n/\ln q$, we have

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

Thus,

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

where k_i are some constants. Since x appears only in the logarithm, we can consider the leading term of Equation (17) approximating $x \approx a\lambda + b$, where a, b are some constants. Thus, using couple optimization, we obtain

$$n \approx \left(\frac{\lambda + A'\ln(\ln q)}{B'\ln(\lambda) + C'} + D'\right)\ln q, \tag{23}$$

$$\begin{array}{lllll} A' = -1.142080 & B' = 0.231197 & C' = 1.106616 & D' = -0.233138 & \text{if } \chi_s = \mathcal{U}_2 \\ A' = -1.073049 & B = 0.278319 & C' = 0.931202 & D' = 0.792882 & \text{if } \chi_s = \mathcal{U}_3. \end{array}$$

The comparison results between the output of the Lattice Estimator and our formula (Equation (23)) are presented in Tables 6 and 7, demonstrating the effectiveness of our approach in accurately estimating security levels.

$\log q$	Est_λ	Est_n	(22)	(23)	$\log q$	Est_λ	Est_n	(22)	(23)
		$\lambda \approx 80$					$\lambda \approx 100$		
42	80	1024	1036	1040	34	100	1024	1037	1041
58	80	1408	1432	1428	46	102	1408	1427	1429
71	80	1728	1754	1743	57	100	1728	1735	1734
84	80	2048	2076	2057	67	100	2048	2039	2035
		$\lambda \approx 110$					$\lambda \approx 120$		
31	110	1024	1036	1038	28	123	1024	1042	1042
42	112	1408	1424	1426	39	121	1408	1423	1425
52	111	1792	1746	1747	48	121	1792	1748	1750
61	112	2048	2064	2063	57	121	2048	2072	2074
		$\lambda \approx 128$					$\lambda \approx 140$		
27	128	1024	1045	1043	24	144	1024	1042	1034
37	128	1408	1424	1425	34	140	1408	1425	1424
45	129	1728	1739	1742	41	143	1728	1746	1748
54	128	2048	2068	2072	49	142	2048	2068	2073

Table 6: Comparison between the LWE dimension provided by our formula Equation (23) 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	(22)	(23)	$\log q$	Est_λ	(22)	(23)
43	80	1054	1082	1400	80	34247	33534
34	102	1056	1066	1100	101	33420	32910
32	109	1060	1065	1000	112	33422	32971
29	122	1074	1069	930	121	33369	32959
27	132	1081	1068	880	128	33244	32863
25	143	1084	1062	810	140	33206	32870

Table 7: Comparison between the LWE dimension provided by our formula Equation (23) 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 5, we pictured the data points of the Lattice Estimator for uSVP attack and our formula proposed in Equation (23).

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

Starting from Equation (9), and setting $\beta \approx (\lambda - \ln(\lambda))/0.292$, we use the couple optimization to find the resulting function for n (considering the BDD attack):

$$n = \frac{(\tilde{A}\beta + \tilde{B})\left(2\ln q + \ln(\beta/2\pi e) + \tilde{C}\right)^2}{2\left(\ln(\beta/2\pi e) + \tilde{D}\right)\left(\ln(q^2/\zeta)\right)^2} \ln q \tag{24}$$

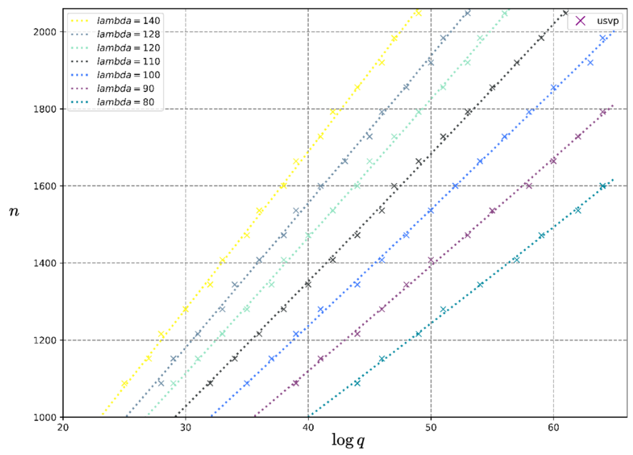


Fig. 5: Comparison between Equation (23) and the data points output by the Lattice Estimator for $\chi_s = \mathcal{U}_2$, considering the uSVP attack.

where

$$\begin{array}{llllll}\tilde{A} = 1.154587 & \tilde{B} = -46.18551 & \tilde{C} = -4.457340 & \tilde{D} = 0.809972 & \text{if } \chi_s = \mathcal{U}_2 \\ \tilde{A} = 1.417954 & \tilde{B} = -48.44275 & \tilde{C} = -2.871196 & \tilde{D} = 1.884925 & \text{if } \chi_s = \mathcal{U}_3.\end{array}$$

We approximate Equation (24) obtaining

$$\begin{aligned} n \approx & \frac{(k_1\lambda + k_5 \ln\lambda)\big((\ln q + k_2) + (k_3 \ln\lambda - k_3 \ln\ln\lambda + k_4)\big)^2}{(k_3 \ln\lambda - k_3\ln\ln\lambda + k_4)\ln q} \\ \approx & (k_1\lambda + k_5\ln\lambda)\left(\frac{\ln q + k_6}{k_3\ln\lambda - k_3\ln\ln\lambda + k_4} + \frac{k_3\ln\lambda - k_3\ln\ln\lambda + k_4}{\ln q}\right) \\ \approx & (k_1\lambda + k_5\ln\lambda)\left(k_7\frac{\ln q}{\ln\lambda} + k_8\frac{\ln\lambda}{\ln q}\right) \end{aligned}$$

for some constant $k_i \in \mathbb{R}$. Using coupled optimization techniques we have

$$n = (A'\lambda + B'\ln\lambda)\left(C'\frac{\ln q}{\ln\lambda} + D'\frac{\ln\lambda}{\ln q}\right) \tag{25}$$

$$\begin{array}{llllll}A' = 0.463730 & B' = -1.634159 & C' = 5.236220 & D' = 1.818256 & \text{if } \chi_s = \mathcal{U}_2 \\ A' = 2.755987 & B' = -10.41781 & C' = 0.869780 & D' = 0.318689 & \text{if } \chi_s = \mathcal{U}_3.\end{array}$$

The comparison results between Equations (24) and (25) and the output of the Lattice Estimator are presented in Tables 8 and 9. In Figure 6, we show the data points of the Lattice Estimator for the BDD attack and Equation (25).

$\log q$	Est_λ	Est_n	(24)	(25)	$\log q$	Est_λ	Est_n	(24)	(25)
		$\lambda \approx 80$					$\lambda \approx 100$		
42	80	1024	1047	1050	34	100	1024	1053	1055
58	80	1408	1444	1444	46	102	1408	1445	1445
71	80	1728	1767	1765	57	100	1728	1754	1753
84	80	2048	2090	2087	67	100	2048	2059	2058
		$\lambda \approx 110$					$\lambda \approx 120$		
31	110	1024	1053	1054	28	123	1024	1059	1061
42	112	1408	1442	1442	39	121	1408	1441	1440
52	111	1792	1765	1765	48	121	1792	1766	1766
61	112	2048	2083	2083	57	121	2048	2092	2093
		$\lambda \approx 128$					$\lambda \approx 140$		
27	128	1024	1062	1063	24	144	1024	1059	1060
37	128	1408	1442	1441	34	140	1408	1442	1442
45	129	1728	1758	1758	41	143	1728	1764	1764
54	128	2048	2088	2089	49	142	2048	2085	2088

Table 8: Comparison between the LWE dimension n provided by Equation (25) 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_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	(24)	(25)	$\log q$	Est_λ	(24)	(25)
43	79	1061	1030	1450	77	33555	33716
34	101	1086	1037	1150	97	33272	33292
32	108	1096	1042	1050	106	33040	33061
29	120	1109	1046	930	121	33116	33146
27	129	1116	1046	870	129	32872	32907
25	140	1127	1050	810	140	32999	33042

Table 9: Comparison between the LWE dimension n provided by our formulas (Equation (25)) 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.

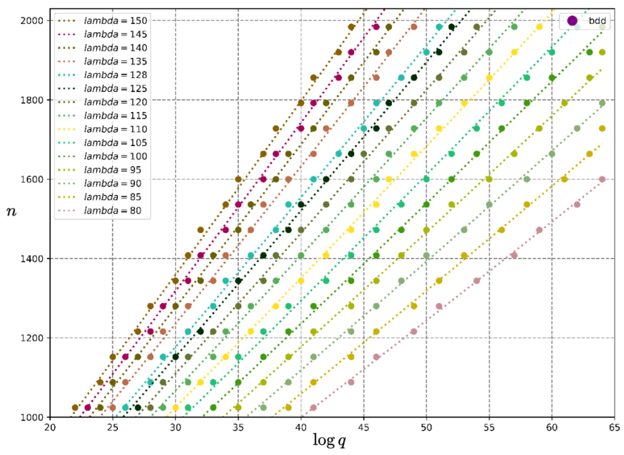


Fig. 6: The LWE dimension n (Equation (25)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$ for the BDD attack.

5 Generalization employing numerical methods

In the previous sections, we derived formulas to express the security level λ and the LWE dimension n from the complexity analysis of the primal BDD and uSVP attacks. The idea is to first approximate the non-leading terms of these equations to retrieve the desired parameter, Section 3, and then compensate with a fine-tuning phase as we presented in Section 4. This approach has several advantages: it is fast, numerically stable, and explicitly shows the relations among the parameters. On the other hand, the fine-tuning phase proposed is specific for $\sigma_e = 3.19$. This choice of the standard deviation of the error distribution is the preferred one for the BGV, BFV, and CKKS schemes. However, in TFHE, it is often required to vary in order to achieve a specific level of security due to the restrictions on the ciphertext size. In the following, we propose a computational alternative that allows to precisely determine the value of any of the parameters λ, n, q, and σ_e (the latter 2 parameters on the ln-scale), provided the remaining ones and the desired secret distribution.
The idea is to employ numerical methods, i.e. mathematical tools designed to solve numerical problems, for the resolution of the systems of equations obtained from the theoretical analysis of the attacks on LWE provided in Section 3. In detail, we get a system of two equations in the two variables β and the desired parameter as follows.

BDD attack. In Section 3.1, we obtained Equations (6) and (8), relating the parameters λ, n, q, σ_s, σ_e and the block size β. Writing them as equations in implicit form, we obtain the following system

$$\begin{cases} \beta - \dfrac{2n \ln q \ln\left(\frac{\beta}{2\pi e}\right)}{\left(\ln\left(\frac{\beta}{2\pi e}\right)+2\ln\left(\frac{q}{\sigma_e\sqrt{2\pi e}}\right)-2\sqrt{\frac{n}{2\ln q}\cdot\frac{\ln\left(\frac{\beta}{2\pi e}\right)}{\beta}\ln\left(\frac{q}{\zeta}\right)}\right)^2} = 0 \\ \lambda - (0.292\beta + \log_2(8d) + 16.4) = 0, \end{cases} \tag{26}$$

where the optimal d is set to $d = \sqrt{\frac{2n \ln q \ \beta}{\ln(\beta/2\pi e)}}$. Note that, in our scenario, we have a system of two equations in two unknowns, the block size β and the parameter to be determined, that we can solve with a numerical method for root finding. In particular, we chose to use Python method `fsolve`, because it appeared fast and effective in practice.

Specifically, in computing the security level λ, we adopt the numerical solver to find β from the first equation. Then plug its value in the second to get λ.

In the computation of n, q and σ_e, we are able to make a further improvement in the precision of our approximation. Indeed, so far we assumed $\beta \approx \eta$, which only introduce little fluctuations in the computation of λ with the numerical approach. However, this approximation can have a bigger impact when estimating other, more delicate parameters like $\ln q$ or $\ln \sigma_e$. Therefore, in this case we use the the version of Equation (6) in which η is not substitute by β. The resulting system of equations is

$$\begin{cases} \eta - d + \frac{1}{\ln \delta_\beta}(\ln \frac{q}{\sigma_e\sqrt{2\pi e}} - \frac{n}{d} \ln \frac{q}{\zeta}) = 0 \\ \lambda - (0.292\beta + \log_2(8d) + 16.4) = 0, \end{cases}$$

and η is computed from $\lambda - (0.292\eta + \log_2(\eta) + 16.4) = 0$, using a numerical method as well.

uSVP attack. Analogously, we can write Equations (11) and (13) from Section 3.2 in a system of two equations describing the relations among the parameters and the block size β,

$$\begin{cases} \beta - \frac{2n\ln(q/\zeta)\ln(\beta/(2\pi e))}{\ln^2(q\sqrt{\beta}/(2\pi e\sigma_e))} = 0 \\ \lambda - (0.292\beta + \ln(8\sqrt{\frac{2n\ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}}) + 16.4) = 0, \end{cases} \tag{27}$$

and find the desired parameter by solving the system with a numerical method.

In our tool, we adopt this strategy to determine the values of the ciphertext modulus q and standard deviation of the error distribution σ_e. Additionally, we propose numerical estimates for the security level λ and the LWE dimension n, comparing the results with the ones obtained using the approach from the previous sections, both for $\sigma_e = 3.19$ and different. To clarify, we do not propose a way to compute σ_s, as the distribution of the secret key is chosen a priori, according to the scheme and the scenario. TFHE-like schemes employ the binary distribution, while BGV, BFV, and CKKS use the ternary if leveled and the sparse if the bootstrapping is expected.

6 How to use our results in practice

In this section we present a powerful tool that integrates the results of Sections 4 and 5. The tool, available in our Github repository[11], allows users to quickly and efficiently select secure LWE parameters.

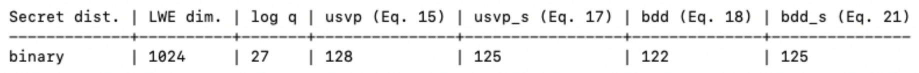
After narrowing down the range of possible parameter choices with our tool, users can verify them using any LWE estimator. For example, one could reply on the Lattice Estimator or on the Leaky-LWE Estimator [25].[12] It is important to note that our tool targets *security* and does not take into account the *correctness* of FHE decryption, since this depends on the circuit being evaluated and the selected FHE scheme.

Now, we will provide a detailed overview of our tool's functionality and demonstrate how to use it in practice.

Estimation of the security level To determine the security level, use the command `-param "lambda"` followed by the known parameters: n, q, σ_s, and σ_e. For example, if you want to know the security of an LWE scheme with $n = 1024$, $\log q = 27$, the binary secret distribution, and the error following a discrete Gaussian distribution with $\sigma_e = 3.19$, you can enter:

```
python3 estimate.py --param "lambda" --n "1024" --logq "27"
   --secret "binary" --error "3.19"
```

which will produce:



```
Secret dist. | LWE dim. | log q | usvp (Eq. 15) | usvp_s (Eq. 17) | bdd (Eq. 18) | bdd_s (Eq. 21)
-------------+----------+-------+---------------+-----------------+--------------+---------------
binary       | 1024     | 27    | 128           | 125             | 122          | 125
```

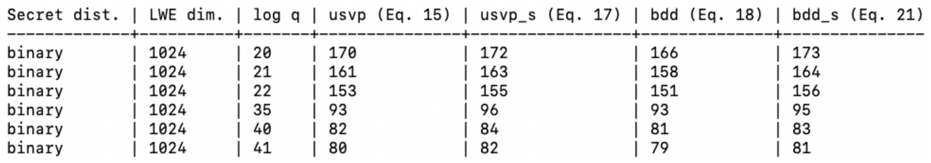
where the `usvp` and `bdd` columns display the outputs of Equations (15) and (18), and the `usvp_s` and `bdd_s` show the results of Equations (17) and (21). It is important to note that the output includes four distinct security level values, each derived from a different approach as previously described. To determine the security level, simply select the minimum value among them; in this example, $\lambda \approx 122$.

In our tool, it is also possible to specify a range for `logq`. For example, to set `logq` with values between 20 and 22, as well as 35 and the range from 40 to 41, you would enter:

```
python3 estimate.py --param "lambda" --n "1024" --logq "20-22;35;40-41"
   --secret "binary" --error "3.19"
```

resulting in:

[11] https://github.com/sergirovira/fastparameterselection
[12] https://github.com/lducas/leaky-LWE-Estimator

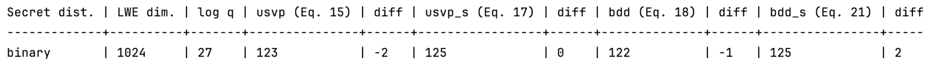


```
Secret dist. | LWE dim. | log q | usvp (Eq. 15) | usvp_s (Eq. 17) | bdd (Eq. 18) | bdd_s (Eq. 21)
-------------+----------+-------+---------------+-----------------+--------------+---------------
binary       | 1024     | 20    | 170           | 172             | 166          | 173
binary       | 1024     | 21    | 161           | 163             | 158          | 164
binary       | 1024     | 22    | 153           | 155             | 151          | 156
binary       | 1024     | 35    | 93            | 96              | 93           | 95
binary       | 1024     | 40    | 82            | 84              | 81           | 83
binary       | 1024     | 41    | 80            | 82              | 79           | 81
```

Finally, you can add the `-verify 1` option to any command to compare the formula's output with the Lattice Estimator's results. For instance, one would enter:

```
python3 estimate.py --param "lambda" --n "1024" --logq "27"
   --secret "binary" --error "3.19" --verify 1
```

which produces:



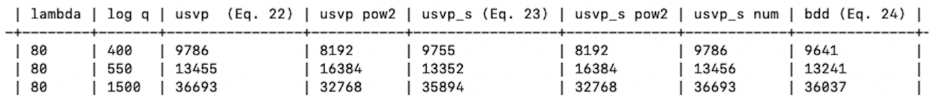
```
Secret dist. | LWE dim. | log q | usvp (Eq. 15) | diff | usvp_s (Eq. 17) | diff | bdd (Eq. 18) | diff | bdd_s (Eq. 21) | diff
-------------+----------+-------+---------------+------+-----------------+------+--------------+------+----------------+-----
binary       | 1024     | 27    | 123           | -2   | 125             | 0    | 122          | -1   | 125            | 2
```

Note that the column `diff` represents the difference between the value in the previous column (i.e., `attack_name`) and the output of the Lattice Estimator run with the parameters specified in the `Secret dist.`, `LWE dim.`, `log q` columns and the value of the previous column. For instance, if we run the Lattice Estimator with $n = 2^{10}$, $q = 27$ and binary secret for the uSVP attack, we obtain a security level of 125 bit. Thus, the value -2 as the difference between the Lattice Estimator's output and the result of our formula Equation (15) `(Eq. 15)`, which is 123.

Estimation of the LWE dimension. The LWE dimension n can be estimated in the same way as explained in the previous paragraph, with the initial command changed to `-param "n"`. Here, one provides the LWE parameters as before, but with target λ replacing n. For instance:

```
python3 estimate.py --param "n" --lambda "80" --logq "400;550;1500"
   --secret "ternary" --error "3.19"
```

A cropped view of the obtained output is:



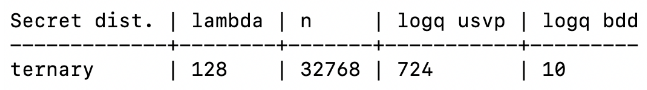
```
 | lambda | log q | usvp  (Eq. 22) | usvp pow2 | usvp_s (Eq. 23) | usvp_s pow2 | usvp_s num | bdd (Eq. 24) |
-+--------+-------+----------------+-----------+-----------------+-------------+------------+--------------+-
 | 80     | 400   | 9786           | 8192      | 9755            | 8192        | 9786       | 9641         |
 | 80     | 550   | 13455          | 16384     | 13352           | 16384       | 13456      | 13241        |
 | 80     | 1500  | 36693          | 32768     | 35894           | 32768       | 36693      | 36037        |
```

where, `attack_name` represents the output of Equations (22) and (24), while the columns `attack_name_s` show the results from the simplified versions of the previous formulas (the ones explicitly dependent on the macro variables n, λ and q), as in Equations (23) and (25). `attack_name pow2` shows the closest power-of-two value corresponding to the n displayed in the previous column, and `attack_name_s num` provides the results computed through the numerical solver (as explained in Section 5). To determine the LWE dimension n for the desired security level, you should select the largest result.

Estimation of the size of the modulus q In this case, the procedure is the same as described earlier, with the only difference being the initial command, which now is `-param "logq"`. Thus, if you enter:

```
python3 estimate.py --param "logq" --lambda "128" --n "32768"
   --secret "ternary" --error "3.19"
```

the output will be the following



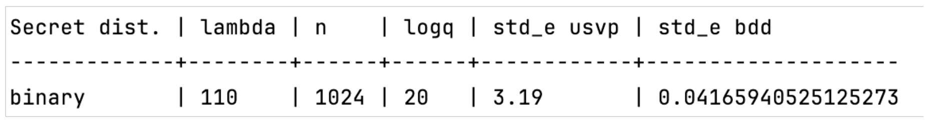
```
Secret dist. | lambda | n     | logq usvp | logq bdd
-------------+--------+-------+-----------+---------
ternary      | 128    | 32768 | 724       | 10
```

Note that the difference here, with respect to the previous solutions, is that the results come only from the numerical method.

Estimation of the standard deviation of the error distribution As before, you can estimate σ_e given the other LWE parameters and the results are again obtained via the numerical method. In this case, the initial command to estimate σ_e is `-param "std_e"`, so if you enter:

```
python3 estimate.py --param "std_e" --lambda "110" --n "1024" --logq "20"
   --secret "binary"
```

The result will be



```
Secret dist. | lambda | n    | logq | std_e usvp | std_e bdd
-------------+--------+------+------+------------+--------------------
binary       | 110    | 1024 | 20   | 3.19       | 0.04165940525125273
```

To determine the standard deviation of the error distribution σ_e for the desired security level, the largest result should be selected, which in this case is $\sigma_e = 3.19$.

7 Advantages of a formula-based approach

Prior to our work, selecting secure parameters for LWE-based FHE schemes was only possible by using the Lattice Estimator [5] and constructing tables based on its outputs. We believe this approach has two major problems: 1) depending on the parameters, the Lattice Estimator can take a long time to produce an output and 2) relying on a set of predefined tables is too rigid, constraining developers and libraries to use those sets of parameters.

The formula-based approach presented in this work solves the previous problems and provides a fast and flexible methodology to select secure parameters for FHE that can directly replace the tables used by existing FHE libraries or provide a faster methodology to update those tables.

Another great advantage of a formula-based approach is total flexibility concerning the parameters that can be fixed. As shown in the previous sections, we provide formulas not only for the security level λ but also for the size of the ciphertext modulus q, the LWE-dimension n and the standard deviation of the error σ_e. This will allow companies working on privacy-preserving applications based on FHE to have total control over the parameters that they need without investing efforts towards constraining their applications to predefined sets of parameters.

In the rest of this section we compare our approach with [13] and [10] which, at the time of writing, are the state-of-the-art approaches for parameter selection in FHE.

7.1 Comparision with [13]

In [13], the authors provide tables listing parameters for FHE applications targeting different levels of security (128, 192 and 256). Moreover, the paper includes a script (based on the Lattice Estimator) which can be used to update the listed values.[13] Their work is particularly valuable to non-experts since it allows them to select secure parameters for their applications quickly.

The main difference between our work and [13] is the scope of parameters that an end-user can obtain. That is, a table-based approach such as the one provided by [13] is rigid by design. Although the authors offer a way to update the parameters via a script, they are restricted to a predefined set of values. The parameters presented in [13] indeed cover most of the current FHE applications but there is no fundamental reason for which we could not obtain parameters outside the usual range. For instance, there might be applications that benefit from a different security level, a smaller LWE dimension or from using a dimension other than a power of two [27]. With our tool, we can quickly get the parameters of Table 10, which would serve the purpose of such applications.

Another difference between our work and [13] is their use of the Lattice Estimator. The script provided by [13] generates the tables by first reading a set of predefined values stored in a lookup table and then runs binary search invoking the Lattice Estimator until optimal parameters are found. Our formula-based approach allows us to obtain optimal values without the need to run the Lattice Estimator, which makes the process of updating the tables much faster. We want to remark that we only use the Lattice Estimator to *verify* and *fine-tune* our formulas while [13] relies on it to produce the tables.

[13] See https://github.com/gong-cr/FHE-Security-Guidelines/

λ	n	$\log q$	χ_s
110	512	17	$\mathcal{U}_3$
128	512	13	$\mathcal{U}_3$
110	3072	101	$\mathcal{U}_3$
128	3072	80	$\mathcal{U}_3$

Table 10: Example of parameters obtained using our tool that are not typically offered in the literature. We selected $\sigma_e = 3.19$ in all the examples.

There are other subtle but important differences between [13] and our work. They use the Matzov attack [52], which is not known to be correct [31], we are relying on more understood attacks. Properly analyzing dual attacks on lattice in the same fashion as we present here for primal attacks (BDD and uSVP) is beyond the scope of this work. Finally, they do not consider sparse secrets nor NTRU while we do not consider quantum attacks.

7.2 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[14], 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 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. This distinction allows us to provide a more general and theoretically grounded parameter selection framework.

8 Conclusion

Starting from a theoretical base, 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. 'Reversing' these formulas we can express any fixed LWE parameter n, or q, or σ_s, or σ_e as a function of the other parameters and the security level λ. We have then verified and fine-tuned our formulas using empirical data obtained from the Lattice Estimator [5]. Additionally, we introduce the use of a numerical method that allows us to precisely determine not only the values of λ and n but also the

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

value of either the (maximal) modulus q or the standard deviation of the error distribution σ_e, given the other LWE parameters.

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 and seeking for a fast, user-friendly and accessible mechanism to choose secure parameters.

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] Bossuat, J., Cammarota, R., Cheon, J.H., Chillotti, I., Curtis, B.R., Dai, W., Gong, H., Hales, E., Kim, D., Kumara, B., Lee, C., Lu, X., Maple, C.,

Pedrouzo-Ulloa, A., Player, R., Lopez, L.A.R., Song, Y., Yhee, D., Yildiz, B.: Security guidelines for implementing homomorphic encryption. Cryptology ePrint Archive (2024)

[14] Brakerski, Z.: Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In: Advances in Cryptology – CRYPTO 2012. pp. 868–886 (2012)

[15] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)

[16] 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)

[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] Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: LWE with side information: Attacks and concrete security estimation. pp. 329–358 (2020). https://doi.org/10.1007/978-3-030-56880-$1_1$2

[26] 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)

[27] Di Giusto, A., Marcolla, C.: Breaking the power-of-two barrier: noise estimation for BGV in NTT-friendly rings. Designs, Codes and Cryptography (Nov 2024). https://doi.org/10.1007/s10623-024-01524-5, https://doi.org/10.1007/s10623-024-01524-5

[28] Ducas, L., Micciancio, D.: FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. pp. 617–640. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)

[29] Ducas, L., Pulles, L.N.: Accurate score prediction for dual-sieve attacks. Cryptology ePrint Archive, Report 2023/1850 (2023)

[30] 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)

[31] Ducas, L., Pulles, L.N.: Does the dual-sieve attack on learning with errors even work? pp. 37–69 (2023). https://doi.org/10.1007/978-3-031-38548-3_2

[32] Ducas, L., van Woerden, W.P.J.: NTRU fatigue: How stretched is overstretched? pp. 3–32 (2021). https://doi.org/10.1007/978-3-030-92068-5_1

[33] 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)

[34] van Elsloo, T., Patrini, G., Ivey-Law, H.: SEALion: A framework for neural network inference on encrypted data. arXiv preprint arXiv:1904.12840 (2019)

[35] Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive (2012)

[36] 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)

[37] Gentry, C.: A fully homomorphic encryption scheme, vol. 20. Stanford university Stanford (2009)

[38] 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)

[39] Hanrot, G., Pujol, X., Stehlé, D.: Analyzing blockwise lattice algorithms using dynamical systems. In: Advances in Cryptology – CRYPTO 2011. pp. 447–464 (2011)

[40] Herold, G., Kirshanova, E., May, A.: On the asymptotic complexity of solving lwe. Des. Codes Cryptography **86**(1), 55–83 (jan 2018)

[41] Hoorfar, A., Hassani, M.: Inequalities on the lambert w function and hyperpower function. J. Inequal. Pure and Appl. Math **9**(2), 5–9 (2008)

[42] 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)

[43] 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)

[44] Kirshanova, E., Marcolla, C., Rovira, S.: Guidance for efficient selection of secure parameters for fully homomorphic encryption. In: International Conference on Cryptology in Africa. pp. 376–400. Springer (2024)

[45] Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Des. Codes Cryptography **75**(3), 565–599 (jun 2015)

[46] Lattigo: http://github.com/ldsec/lattigo

[47] Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: Cryptographers' Track at the RSA Conference. pp. 293–309. Springer (2013)

[48] 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)

[49] 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)

[50] 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)

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

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

[53] 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)

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

[55] Pouly, A., Shen, Y.: Provable dual attacks on learning with errors. pp. 256–285 (2024). https://doi.org/10.1007/978-3-031-58754-2$_1$0

[56] 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)

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

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

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

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

[61] Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: International Conference on the The-

ory and Application of Cryptology and Information Security. pp. 617–635. Springer (2009)

[62] 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)