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

Beatrice Biasioli[1], Elena Kirshanova[1], Chiara Marcolla[1], and Sergi Rovira[1]

Technology Innovation Institute, Abu Dhabi, United Arab Emirates
beatrice.biasioli@ibm.com elenakirshanova@gmail.com
chiara.marcolla@gmail.com roviracisternasergi@gmail.com

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 Learning with Errors (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 relations 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 and other LWE-based schemes in real-world applications, ensuring that our approach is both rigorous and efficient.

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

1 Introduction

With the advancements of future-generation networking technologies like cloud services, artificial intelligence applications, and Internet of Things, 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 [MSM+22].

The first FHE scheme was introduced in 2009 by Gentry [Gen09]. 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 [BGV14], BFV [Bra12, FV12], FHEW [DM15], TFHE [CGGI16, CGGI20], and CKKS [CKKS17, CHK+18]. More details on FHE and its applications can be found in surveys [MSM17, AAUC18, MSM+22].

The security of currently known (practical) FHE schemes is based on the presumed intractability of the (decision) Learning with Errors (LWE) problem, [Reg05], and its ring variant (RLWE) [LPR10]. Informally, the decisional version of LWE consists in distinguishing equations $\{(\mathbf{a}_i, b_i = \mathbf{s} \cdot \mathbf{a}_i + e_i)\}_i \mod q$, perturbed by small noise e_i (also called error), from uniform random tuples from $\mathbb{Z}_q^n \times \mathbb{Z}_q$[1]. The errors e_i's are drawn from a narrow distribution of standard deviation σ_e, while the coordinates of the secret $\mathbf{s}$ are drawn from a distribution with standard deviation σ_s.

The problem arising from lattice-based constructions is that the error grows whenever a homomorphic operation is performed. In particular, in the worst-case scenario, it grows exponentially when homomorphic multiplications are computed. However, in order to guarantee correct decryption, the error has to be small relative to the modulus q. One approach to accommodating more operations is to increase the modulus. 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 λ, which comes at the cost of efficiency.

This 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 [ACC+18] (using the Lattice Estimator[2]) provides upper bounds on the size of the modulus q for given security levels λ and dimensions n through lookup tables, recently updated in [BCC+24]. Moreover, in [MML+23], 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 [KMR24], 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

[1] 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.

[2] The Lattice Estimator (`https://github.com/malb/lattice-estimator` [APS15]) 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.

distribution χ_s, n, and the modulus q, 2) determining n as a function of λ, $\log q$, and χ_s. Our work is based on [KMR24], which we significantly 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 [VJH21, MSM+22]. Some examples are ALCHEMY [CPS18], Cingulata [CDS15], EVA [DKS+20], HEIR [Con23] and SEALion [vEPIL19]. Additionally, Bergerat *et al.* [BBB+23] proposed a framework for efficiently selecting parameters in TFHE-like schemes. In [MML+23], the authors developed an interactive parameter generator for the leveled BGV scheme, which supports arbitrary circuit models, and Biasioli *et al.* [BMCM23] further extended this approach to the BFV scheme.

Although these contributions mark significant progress toward improving accessibility 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 [Pai], 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. We extend our previous work [KMR24] in several directions: we express the LWE parameters q and χ_e (in bit size) via the remaining LWE parameters and a given security level λ. Moreover, we test our formulas with a wider range of secret and error distributions (see Sections 2.1 and 6.1). For example, for the error distribution, we support now Gaussian, Ternary, Binary, and Sparse ternary errors. Furthermore, we introduce the use of numerical solvers that allow us to find precise estimates for the cases when obtaining concrete analytical formulas is too cumbersome. Finally, we have greatly enhanced our tool with our new results and have included useful functionalities to help developers find optimal FHE parameters faster.

Our analysis focuses on the following types of lattice algorithms: the unique Shortest Vector Problem (uSVP) attack [Kan83, AGVW17], the Bounded Distance Decoding (BDD) attack [LN13], and the primal hybrid attack [HG07]. We do *not* consider the (heuristic) versions of dual like [MAT22, GJ21] since, at the time of writing, these attacks do not offer correctness [DP23].[3] Investigating the impact of the recent proposal from [CMHSJP25] that corrects the flaw of the analysis of heuristic dual attacks is left for future work.

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

[3] Even if [MAT22] was correct, the improvement over uSVP or BDD would be rather marginal, as can be seen by running the Lattice Estimator. The provable versions of dual attacks [PS24] that come with correctness guarantees are inferior to the attacks considered here for concrete parameters.

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 bit size of the modulus q.
   - express the LWE dimension n in terms of λ, $\log q$, σ_e, and σ_s,
   - express the bit size of the LWE modulus q in terms of λ, n, σ_e, and σ_s,
   - express the bit size of the standard deviation of the LWE noise σ_e in terms of λ, q, n, and σ_s.
2. Using our formulas as a starting point, we build more precise estimates by conducting extensive experiments with the Lattice Estimator [APS15], creating a large dataset that correlates $n, \log q, \sigma_e$, σ_s, and λ and producing a fitting function that relates the FHE-inspired 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 express any LWE parameter as a function of the other LWE parameters.
4. More importantly for practitioners, we provide a practical tool where we implement our formulas. Written in Python and publicly available on Github repository[4], 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 [ABD16, KF17, DvW21].

Comparison with related work. In [MML+23], the authors empirically derived and fine-tuned a formula linking the security level λ with the LWE dimension n for a given modulus q and secret distribution. The main difference between the formula provided in [MML+23] and our formulas lies in the level of specificity. In this work, we provide distinct formulas for various attacks against the LWE problem. In contrast, the authors of [MML+23] propose a single generic formula that approximates the behavior of all the attacks for ternary (or Gaussian) secret. This comes at the cost of accuracy, especially because the complexity and best algorithms change depending on the sparsity of the secret, which is hard to capture by a generic formula.

In [BBB+23], 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.

[4] https://github.com/Crypto-TII/fastparameterselection

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 [BBB+23] uses the fitting function to build the totality of their formula, our use is solely for optimizing lower-order terms.

Finally, in [BCC+24], 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 [BCC+24] is the scope of parameters that an end-user can obtain. That is, a table-based approach such as the one provided by [BCC+24] 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. A more detailed comparison with related work is provided in Section 7.

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 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 [Lat, BBB+22, SEA19]. Using a formula-based method, we get the best of both approaches. Namely, we can get close to optimal parameters for any given application instantly. Another advantage of using formulas is that we can understand the behavior 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, see Section 6.1 for some examples.

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 primitives in real-world applications.

The structure of the paper is as follows: Section 2 introduces the notations and mathematical background necessary for understanding the paper. In Section 3, we provide a comprehensive analysis of the uSVP, BDD and hybrid attacks, deriving formulas that establish the relationships among the 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 $\mathbf{x}$, $\mathbf{x}_i$ denotes the i-th scalar component of the vector. 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 $\mathbf{A}$. For a vector $\mathbf{x} \in \mathbb{R}^n$, denote by $\mathbf{x}_{a:b}$ for $0 \leq a \leq b \leq n$ the coordinates of $\mathbf{x}$ indexed from a (inclusive) to b (exclusive). The position a (resp. b) is dropped if $a = 0$ (resp. $b = n - 1$). The notation extends to matrices column-wise. By log we denote the base-2 logarithm.

Let χ be a probabilistic distribution and $a \in \mathbb{R}$, we write $a \leftarrow \chi$ when sampling a from χ. We use the following distributions.

- Uniform binary distribution $\mathcal{U}_2$ over the set $\{1, 0\}$.
- Uniform ternary distribution $\mathcal{U}_3$ over the set $\{0, \pm 1\}$.
- Uniform modulus distribution $\mathcal{U}_p$ over $\mathbb{Z}_p$, where p is a positive integer.
- Uniform distribution $\mathcal{U}_{[a,b]}$ over a real interval $[a, b] \subset \mathbb{R}$.
- Discrete Gaussian distribution $\mathcal{DG}(0, \sigma^2)$, centered in 0 with standard deviation σ.
- The centered binomial distribution ψ_η of width $\eta \in \mathbb{N}$ chooses 2η uniform independent bits $a_i, b_i \in \{0, 1\}$ and computes $\sum_{i=1}^{\eta}(b_i - a_i)$.
- Sparse ternary distribution $\mathcal{HWT}(h)$ chooses a vector uniformly at random from $\{0, \pm 1\}^n$ with exactly h nonzero entries, where $h \leq n$ positive integer.

2.2 Mathematical background

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

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

Such $\mathbf{B}$ is called a basis of $\mathcal{L}$. If $k = n$, the lattice is said to be *full rank*. We will be concerned with integral lattices, 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 *unique SVP (uSVP)*, we are guaranteed that the first successive minimum is $\gamma > 1$ times smaller than the second minimum $\lambda_2(\mathcal{L})$. 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 on a lattice. 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[5] σ 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, [Sch87]). Together with a lattice basis, it receives as input an 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

[5] 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 $\sigma > 0.6$.

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 [HPS11] it is shown that after poly(n) 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 [BDGL16] that outputs a shortest vector in an n-dimensional lattice in time $2^{0.292n+o(n)}$. We choose this running time (ignoring the $o()$-term) as the measure of SVP hardness. Further, for a more concrete complexity of BKZ-β on an n-dimensional lattice we set the running time of BKZ as

$$T_{\mathrm{BKZ}}(\beta, n) = 2^{0.292\beta + \log_2(8n) + 16.4}, \tag{3}$$

which is the choice adopted by [BDK+18, FHK+18, DKL+18]. The correcting constant of 16.4 was obtained experimentally [BDGL16]. The concrete choice of $T_{\mathrm{BKZ}}(\beta, n)$ is called the core-SVP model [ADPS16]. Our results are easy to adapt to other similar choices of $T_{\mathrm{BKZ}}(\beta, n)$.

In addition to Equation (1), BKZ quality guarantees extend (heuristically) to norms of Gram-Schmidt vectors of the returned basis. It is formulated in Geometric Series Assumption. All known lattice estimators [APS15, DSDGR20] rely on this assumption.

Definition 1 (Geometric Series Assumption (GSA), [Sch03]). *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 the attacks considered in this work, we need an efficient BDD solver: Babai's algorithm [Bab86]. 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 the BDD error vector $\mathbf{t} - \mathbf{v}$ 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 [HKM18].[6]

Lemma 1 ([HKM18, 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 [Reg05]. 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 $\chi_s \in \{\mathcal{U}_3, \mathcal{U}_2, \mathcal{HWT}(h)\}$. For the error, we are concerned with discrete Gaussian distribution centered at 0 with standard deviation σ_e. In particular, for BGV, BFV and CKKS, $\sigma_e = 3.19$ [ACC+18], instead for TFHE parameters [CGGI16, CGGI20], σ_e may vary, and we consider these regimes too.

There exist several versions of LWE: Ring-LWE [SSTX09, LPR10] and Module-LWE [LS15]. 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 (uSVP and BDD), and do not consider the dual attacks. First, the recent discoveries [DP23] of failing heuristics employed in efficient dual attacks [GJ21, MAT22] invalidate the claimed complexities. Despite of ongoing attempts to bring dual attacks back

[6] Even though in [HKM18, Lemma 4] the authors talk about *continuous* Gaussian, the result holds for the discrete Gaussian too.

into play [DP23], 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 also consider here the so-called *hybrid* attacks [HG07, Alb17]. These are relevant for sparse secret LWE, i.e., for cases when the Hamming weight of the secret is less than $n/2$.

We receive as input an LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, 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 three attacks: uSVP, BDD, and hybrid, and derive accurate formulas for their complexities. For uSVP and BDD we reverse these formulas to express n as a function of $\log q, \sigma_e, \sigma_s$, and the desired security level λ; analogously, we express $\log q$ and σ_e as functions of the other LWE parameters reaching the given security level. For hybrid, we also provide formulas for the complexity of the attack as functions of LWE parameters, and use these formulas to derive $\log q$ for the given security level.

3.1 The uSVP attack

One way to evaluate the hardness of LWE is to model it as the problem of finding a unique shortest vector (uSVP) in a lattice. Concretely, consider the following lattice

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

commonly referred to as Kannan's embedding lattice [Kan83], where $d = m+n$. The lattice $\mathcal{L}_{\texttt{uSVP}}$ admits the following basis matrix (written column-wise):

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

The constant 1 in the lattice basis is a conventional choice [AGVW17] and can be adjusted for concrete parameters.

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

$$\mathbf{B}_{\texttt{uSVP}} \cdot (\mathbf{s}, -\mathbf{k}, 1)^t = (\mathbf{s}, \mathbf{e}, -1)^t \in \mathcal{L}_{\texttt{uSVP}},$$

and, for typical LWE parameters, $(\mathbf{s}, \mathbf{e}, 1)$ is 'unexpectedly' short. Concretely, we have $\|(\mathbf{s}, \mathbf{e}, 1)\| \approx \sqrt{n\sigma_s^2 + m\sigma_e^2 + 1}$. In cases where $\sigma_s < \sigma_e$, one can 're-balance' the contribution of $\mathbf{s}$ and $\mathbf{e}$ to the norm by scaling the $\mathbf{I}_n$ part of $\mathbf{B}_{\texttt{uSVP}}$ by $\zeta = \max\{1, \lfloor \sigma_e/\sigma_s \rceil\}$:

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

Even though $\zeta > 1$ increases the length of the shortest vector in $\mathcal{L}_{\mathtt{uSVP}}$, at the same time it scales $\det(B_{\mathtt{uSVP}})$ by a factor ζ^n, which is beneficial for lattice reduction.

The primal uSVP attack runs BKZ reduction [Sch87, SE94] on $B_{\mathtt{uSVP}}$ in order to find the unique (up to sign) shortest vector. The estimates [ADPS16, AGVW17] predict that BKZ succeeds in finding $(\zeta\mathbf{s}|\mathbf{e}|1)$ if

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

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

$$\sqrt{\beta}\sigma_e \leq \delta_\beta^{2\beta-d} \cdot \zeta^{\frac{n}{d}} \cdot q^{1-\frac{n}{d}}. \tag{4}$$

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 (6) for the final result.

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 the attack is), we aim at finding m that maximizes the right-hand side of Inequality (4). The maximum is achieved for $d = \sqrt{\frac{n\ln(q/\zeta)}{\ln\delta_\beta}}$. Substituting it in the Inequality (4) and taking logarithms leads to the success condition:

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

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

$$\beta \geq \frac{2n\ln(q/\zeta)\ln(\beta/(2\pi e))}{\ln^2(q\sqrt{\beta}/(2\pi e\sigma_e))}. \tag{5}$$

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 (the equation below is rather the inequality giving the lower bound on successful β):

$$\beta = \frac{2n\ln(q/\zeta)\ln\left(\frac{n\ln(n/\ln q)}{2\pi e\ln(q/\sigma_e)}\right)}{\ln^2\left(\frac{q\sqrt{n\ln(n/\ln(q/\sigma_e))/\ln q}}{2\pi e\sigma_e}\right)} \tag{6}$$

Substituting Equation (6), obtain the expression for λ

$$\lambda = 0.292\beta + \log_2\left(8\sqrt{\frac{2n\ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}}\right) + 16.4. \tag{7}$$

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 Inequality (5) 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

$$n = \frac{\lambda\left(0.5\ln(\lambda/0.292) + \ln(q/(2\pi e\sigma_e))\right)^2}{0.584\ln(q/\zeta)\ln(\lambda/(0.584\pi e))}. \tag{8}$$

We defer from refinements of the expression as they involve tedious computations coming from a more accurate expression of β, Equation (3). Later we show that Equation (8) already provides a very good accuracy.

Expressing $\ln q$. We notice that Inequality (5) 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 $\beta \approx \lambda/0.292$, we obtain the expression for $\ln q$ as a function of LWE parameters and λ.

Expressing $\ln \sigma_e$. Similarly, Inequality (5) is quadratic in $\ln \sigma_e$. In case $\zeta = \sigma_e/\sigma_s$, the solution that is relevant for us 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

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

3.2 The BDD attack

While the BDD attack on LWE has been known for years [LN13], we did not find a reference that aligns well with the Lattice Estimator [APS15], 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

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

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 very similar to $\mathbf{B}_{\mathtt{usvp}}$:

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

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

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

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

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

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

If $\sigma_s < \sigma_e$, one can again 're-balance' the contribution of $(\mathbf{s}, -\mathbf{e})$ into the distance $\sqrt{n\sigma_s^2 + m\sigma_e^2}$ 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 [LN13] 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

$$\|\pi_{d-\eta+1}((\mathbf{s}, -\mathbf{e}))\| < \lambda_1(\mathbf{B}'_{\mathtt{bdd},[d-\eta+1,d]})$$

for $0 \leq \eta < d$ as small as possible. Under the Gaussian Heuristic and the approximation $\|\pi_{d-\eta+1}((\mathbf{s}, \mathbf{e}))\| \approx \sigma_e\sqrt{\eta}$, the above inequality can be rewritten as

$$\sigma_e\sqrt{\eta} < \frac{\sqrt{\eta}}{\sqrt{2\pi e}} \det\left(\mathbf{B}'_{\mathtt{bdd},[d-\eta+1,d]}\right)^{1/\eta}. \tag{9}$$

This condition means that the orthogonal projection of our short vector $(\mathbf{s}, \mathbf{e})$ on $\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1, \ldots, \mathbf{b}_{d-\eta+1})$ is shorter than the shortest vector in the projected lattice $\mathbf{B}'_{\mathtt{bdd},[d-\eta+1,d]}$ given by the basis $(\pi_{d-\eta+1}(\mathbf{b}'_{d-\eta+1}), \ldots, \pi_{d-\eta+1}(\mathbf{b}'_d))$. In the LWE setting, GSA suggests that for small η's the left-hand side of Ineq. (9) 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. (9) 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-$(\eta+1)$ 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 $\mathbf{B}'_{\mathtt{bdd},[1,d-\eta]}$, which is a sublattice of $\mathbf{B}'_{\mathtt{bdd}}$ generated by its first $(d-\eta)$ vectors. The norms of Gram-Schmidt vectors of this sublattice, $\|\mathbf{b}_1^\star\|,\dots\|\mathbf{b}_{d-\eta}^\star\|$ satisfy

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

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. (9). Applying Lemma 1 to $\mathbf{B}'_{\mathtt{bdd},[1,d-\eta]}$ gives constant probability of Babai algorithm to output $(\mathbf{s},\mathbf{e})$.

Runtime analysis of BDD. Let us now analyse the runtime of this attack. Among the three steps of the BDD attack, the most expensive ones are the first step (BKZ-β) and the second (SVP in dimension η). It is optimal to balance these two steps.

The runtime of BKZ-β on a d-dimensional lattice as given in Equation (3) is $T_{\mathrm{BKZ}}(\beta,d) \approx 2^{0.292\beta}\cdot 8d$, while the runtime of SVP on η-dimensional lattice is $T_{\mathrm{SVP}}(\eta) = 2^{0.292\eta}$. The two runtimes differ only by 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. (9). Concretely, using GSA and the BKZ-β guarantee on $\|\mathbf{b}'_1\|$, we compute

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

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. Maximizing the expression $\delta_\beta^{-\eta(d-\eta-1)}\cdot(q^m\zeta^n)^{\frac{\eta}{d}}$ with respect to d, yields optimal lattice dimension $d=\sqrt{\frac{n\ln(q/\zeta)}{\ln\delta_\beta}}$. From Ineq. (9) and Equation (2), we obtain the following expression for β as a function of d,q,σ_e,ζ:

$$\beta \geq \frac{d\ln\left(\frac{\beta}{2\pi e}\right)}{\ln\left(\frac{\beta}{2\pi e}\right)+2\ln\left(\frac{q}{\sigma_e\sqrt{2\pi e}}\right)-2\frac{n}{d}\ln\left(\frac{q}{\zeta}\right)}. \tag{10}$$

Substituting the optimal choice for d in the equation above yields

$$\frac{\beta}{\ln\left(\frac{\beta}{2\pi e}\right)} \geq \frac{2n\ln q}{\left(\ln\left(\frac{\beta}{2\pi e}\right)+2\ln\left(\frac{q}{\sigma_e\sqrt{2\pi e}}\right)-2\sqrt{\frac{n}{2\ln q}\cdot\frac{\ln\left(\frac{\beta}{2\pi e}\right)}{\beta}}\ln\left(\frac{q}{\zeta}\right)\right)^2} \tag{11}$$

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 (11). 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 (11) 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[7], which can be evaluated numerically for our parameters. Concretely, we obtain $\beta = 2\pi e^{1-W_1\left(-\frac{2\pi eB}{2D\sqrt{C}+\sqrt{A}}\right)}$, where $W_1()$ denotes the "lower" branch of Lambert-W function. It follows $\ln\left(\frac{\beta}{2\pi e}\right) = -W_1\left(-\frac{2\pi eB}{2D\sqrt{C}+\sqrt{A}}\right)$. Substituting this result in Equation (11), we obtain a closed expression for β (technically, it is a lower bound for β, but we treat it as equality):

$$\beta = \frac{2n\ln q \cdot \left(-W_1\left(-\frac{2\pi eB}{2D\sqrt{C}+\sqrt{A}}\right)\right)}{\left(-W_1\left(-\frac{2\pi eB}{2D\sqrt{C}+\sqrt{A}}\right) + 2\ln\left(\frac{q}{\sigma_e\sqrt{2\pi e}}\right) - \sqrt{\frac{n}{2\ln q}} \cdot \frac{B}{2D\sqrt{C}+\sqrt{A}} \ln(q/\zeta)\right)^2} \tag{12}$$

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:

$$\lambda = \log(T_{\mathrm{BKZ}}(\beta, d), 2) = 0.292\beta + \log_2(8d) + 16.4. \tag{13}$$

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 (10). 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

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

$n, \lambda, q, \sigma_e, \sigma_s$. 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

$$n = \frac{2\ln q \cdot (\lambda - \log(8d)) \cdot (2\ln q + \ln((\lambda - \log(8d))/(0.584\pi e)) - 2\ln(\sigma_e\sqrt{2\pi e}))^2}{0.292\ln((\lambda - \log(8d))/(0.584\pi e))(4\ln q - 2\ln\zeta)^2}. \tag{14}$$

Expressing $\ln q$. Inspecting Equation (10), we notice that it is linear in $\ln q$ inequality (treated here as equality). Concretely,

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

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 $\lambda, n, \sigma_e, \sigma_s$.

Expressing $\ln\sigma_e$. Similarly to $\ln q$, a closer look at Equation (10) 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. Instead we resort to numerical computations (see later in Section 5) to compute σ_e.

3.3 Hybrid attack

Some FHE schemes [CKKS17] employ LWE instances with *sparse* ternary secrets, i.e., secrets with few non-zero coordinates. In this regime, a hybrid attack may perform better than BDD or uSVP. The hybrid attack [LN13, Alb17, Ber23] is a generalization of the BDD attack, where we start by guessing random n_g coordinates of the secret $\mathbf{s}$ for a fixed $0 \le n_g \le n$. Call the guess $\mathbf{s}_g$. If the guess is correct, that is $\mathbf{s}_g = \mathbf{s}_{:n_g}$, then we can reduce the dimension of the LWE instance to $n - n_g$ since

$$\mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{b} \bmod q \iff \mathbf{A}_{n_g:}\mathbf{s}_{n_g:} + \mathbf{e} = \mathbf{b} - \mathbf{A}_{:n_g}\mathbf{s}_{:n_g}. \tag{15}$$

For the correct guess $\mathbf{s}_{:n_g}$, the right-hand side forms a valid BDD instance on the lattice

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

with the target vector $(\mathbf{0}, \mathbf{b} - \mathbf{A}_{:n_g}\mathbf{s}_{:n_g}) \in \mathbb{Z}^{n-n_g} \times \mathbb{Z}^m$, thus we can solve it using lattice reduction as described in Section 3.2. If $\mathbf{s}_{n_g:}$ is a wrong guess, then Equation (15) does not hold, and we do not have any BDD guarantee. Deciding whether a guess is correct or not is implemented by running a BDD attack on each guess. Thus, we arrive at a trade-off between the total number of guesses $\mathbf{s}_g$ we make and the running time of the BDD attack on a lattice of dimension $d := n - n_g + m$.

Let us focus on the case of ternary secrets of Hamming weight h with $h/2$ ones and $h/2$ minus ones. This is the most common choice in the FHE literature, see for example [CH18, GV23, LW24, AKP24]. Fix two integer parameters: n_g – the number of coordinates we guess, and $0 \leq \omega \leq \min(n_g, h)$ – the Hamming weight of $\mathbf{s}_g \in \mathbb{Z}^{n_g}$'s we are guessing. The total number of guesses, that is, the number of ternary vectors of dimension n_g on weight ω is

$$T_{\text{search}}(n_g, w) = \binom{n_g}{\omega} \cdot 2^{\omega}.$$

The probability that the LWE secret $\mathbf{s}$ indeed has weight ω on n_g coordinates is

$$p_{\text{succ}}(n_g, w) = \frac{\binom{n-h}{n_g-w}\binom{h}{w}}{\binom{n}{n_g}}, \tag{16}$$

where the denominator enumerates the total number of n_g-dimensional subvectors of $\mathbf{s}$, and the nominator computes the number of choices for zero's in $\mathbf{s}_g$ times the number of choices for non-zero's in $\mathbf{s}_g$.[8]

The hybrid attack starts by running a BKZ-β lattice reduction on $\mathcal{L}_{\texttt{hybrid}} \in \mathbb{Z}^d$ for some parameter β that guarantees that the last Gram-Schmidt vector $\mathbf{b}_d^\star$ is larger than the standard deviation of the error. It essentially means (up to $\sim (\log n)^{1/2}$, see Lemma 1) that Babai's algorithm will succeed in lifting $\pi_{d-\beta-1}(\mathbf{s}_{n_g:}, \mathbf{e})$ to $\mathcal{L}_{\texttt{hybrid}}$ with constant probability. Under GSA, it means that (on the $\log_2$-scale):

$$(-d+1)\log_2(\delta_\beta) + \frac{1}{d}\left((d-n+n_g-1)\cdot \log_2 q + (n-n_g)\log_2(\zeta)\right) \geq 2\log(\sigma_e^2), \tag{17}$$

where the left-hand side is $\log_2 \|\mathbf{b}_d^\star\|^2$ under GSA. The above inequality guarantees that the correct guess $\mathbf{s}_g$ will be identified by a BDD solver on input

[8] Instead of guessing the exact weight of $\mathbf{s}_g$, one can guess that the secret is *up to* weight ω. This will increase the success probability to $\sum_{i=1}^{\omega} p_{\text{succ}}(n_g, i)$ but also increase the search space to $\sum_{i=1}^{\omega} T_{\text{search}}(n_g, i)$. Asymptotically, the largest term in both of these sums is for $i = \omega$ since $\omega < h \ll n_g/2$.

$(\mathcal{L}_{\texttt{hybrid}}, (\mathbf{0}, \mathbf{b} - \mathbf{A}_{:n_g}\mathbf{s}_{:n_g}))$. Overall, the runtime of the hybrid attack is

$$T_{\text{hybrid}}(\beta, d, n_g, w) = \frac{T_{\text{BKZ}}(\beta, d) + T_{\text{search}}(n_g, w)}{p_{\text{succ}}(n_g, w)}. \tag{18}$$

Note that there are four parameters to optimize: β – the block-size for BKZ lattice reduction, $d = n - n_g + m$ – the dimension of $\mathcal{L}_{\texttt{hybrid}}$, n_g – the number of coordinates of the guessed secret, and finally, ω – the Hamming weight of the guessed secret. Given the target security level λ, these parameters are related as follows:

$$0.292\beta + \log_2(8d) + 16.4 = \log_2\left(\binom{n_g}{\omega} \cdot 2^{\omega}\right), \tag{19}$$

$$d - n_g = \left\lceil \sqrt{\frac{n \ln q}{\ln \delta_\beta}} \right\rceil, \tag{20}$$

where Equation (19) balances (on the $\log_2$ scale) the total number of guesses with the runtime of BKZ, and Equation (20) gives the optimal dimension of the lattice analogously to the case of the BDD attack. Together with Equation (17) (treated as equality), we have three relations between the four optimization parameters. Typically, the Hamming weight ω is a small constant, and, in practice, it is easy to brute-force over it. The shapes of Equations (17), (19) and (20) do not appear to be amenable to a nice analytic expression for neither of the LWE parameters, hence we suggest deriving these parameters using a numerical solver. The details are given in Section 5.

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 one parameter as a function of the rest (for example, the security parameter λ as a function of LWE parameters or the LWE dimension n via $\lambda, \log q, \sigma_s, \sigma_e$). 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 [APS15] and a fitting function. We remark that the simplifications only have a noticeable effect on the non-leading terms and 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 $\chi_s \in \{\mathcal{U}_2, \mathcal{U}_3\}$, we run the Lattice Estimator to obtain the security level of

the corresponding points. It is worth noticing that $\chi_s = \mathcal{U}_2$ is employed in TFHE-like schemes where $2^{10} \leq n \leq 2^{11}$, while $\chi_s = \mathcal{U}_3$ 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 and adjust our formulas exhaustively. Following common practice in the FHE literature we populate our database with parameters offering at least 80 bits of security [CS16, CS17, MHWW24]. Table 1 shows the number of points that we considered.

χ_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 comparing 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 (7), (8), (13) and (14)) by incorporating additional variables. Using *coupled optimization*[9], we determine the optimal values for these variables to ensure that our parameterized functions follows the data points generated with the Lattice Estimator, i.e., accurately reflects the security level estimation.
2. Finally, we provide a further simplification of these formulas, explicitly depending on the variables n, λ and $\log q$. 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).

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

4.2 Verification of uSVP security level, Equation (7)

Starting from Equation (7) and using the process explained above, the resulting function for λ (considering the uSVP attack) is

$$\lambda = A\beta + B\ln\left(\frac{2n\ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}\right) + C, \tag{21}$$

where

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

Now, our aim is to express Equation (21) in a simplified form that explicitly depends on the variables n and q.

Let $x = n/\ln q$, $k_1 = \frac{1}{2\pi e}$ and $k_2 = \frac{1}{2\pi e\sigma_e} = \frac{k_1}{\sigma_e}$. Since $\ln(q/\zeta) \approx \ln(q/\sigma_e) \approx \ln(q)$, we have that Equation (6) 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

$$\beta \approx 2x\left(\ln(x) + \ln(\ln(x)) - 2.8\right). \tag{22}$$

Substituting Equation (22) in Equation (21) 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 approximation

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

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

The comparison results between the output of the Lattice Estimator and our formulas (Equations (21) and (23)) are presented in Tables 2 and 3, demonstrating the effectiveness of our approach in accurately estimating security levels.

Moreover, we want to point out that although our formulas are fine-tuned for $n = 2^\kappa$, with $\kappa \in \{10, \ldots, 15\}$, they still provide accurate security estimates for larger powers of two, such as $n = 2^{16}$ and $n = 2^{17}$, as demonstrated in Table 3.

$n = 2^{10}$				$n = 2^{11}$			
$\log q$	Estimator	(21)	(23)	$\log q$	Estimator	(21)	(23)
20	172	178	174	37	193	193	188
24	142	145	144	46	152	152	149
25	136	139	137	50	139	139	136
26	130	133	132	53	130	130	128
27	125	128	126	54	127	128	126
28	120	123	122	57	120	121	119
30	112	114	113	62	110	111	109
33	101	103	102	67	101	102	101
37	90	92	90	74	91	93	91
42	80	81	78	84	80	82	80

Table 2: Comparison between the security level provided by our formulas (Equations (21) and (23)) and the Lattice Estimator with with $\chi_s = \mathcal{U}_2$.

$n = 2^{10}$				$n = 2^{15}$			
$\log q$	Estimator	(21)	(23)	$\log q$	Estimator	(21)	(23)
16	231	215	233	650	179	155	180
18	204	187	202	760	150	130	151
19	192	175	190	810	140	121	141
25	143	126	137	880	128	110	128
27	131	115	126	930	120	104	121
28	126	110	121	1000	111	96	112
30	117	102	112	1050	106	91	106
32	109	94	104	1200	92	80	93
43	79	68	76	1400	79	69	80
48	71	60	68	1500	74	64	75

$n = 2^{16}$				$n = 2^{17}$			
$\log q$	Estimator	(21)	(23)	$\log q$	Estimator	(21)	(23)
1776	128	110	128	3576	128	111	128
1229	192	167	193	2469	192	167	193
955	256	224	260	1918	256	224	259

Table 3: Comparison between the security level provided by our formulas (Equations (21) and (23)) and the Lattice Estimator with $\chi_s = \mathcal{U}_3$.

In Figure 1 we plot the data points of the Lattice Estimator and our formula proposed in Equation (23).

4.3 Verification of BDD security level, Equation (13)

Starting from Equation (13) and using couple optimization, the resulting function for λ (considering the BDD attack) is

$$\lambda \approx \tilde{A}\beta + \tilde{B} \ln\left(\frac{2n\beta \ln(q/\zeta)}{\ln(\beta)}\right) + \tilde{C}. \tag{24}$$

where

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

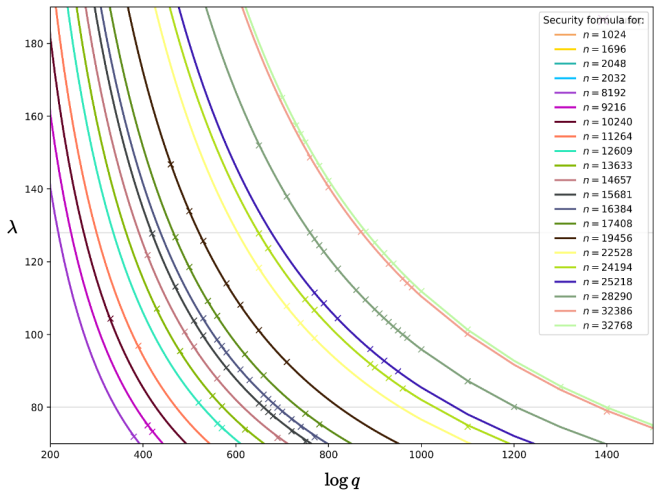

Fig. 1: The security formula (Equation (23)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$ considering the uSVP attack.

In Figures 2 and 3 we pictured the data points of the Lattice Estimator and our formula proposed in Equation (24).

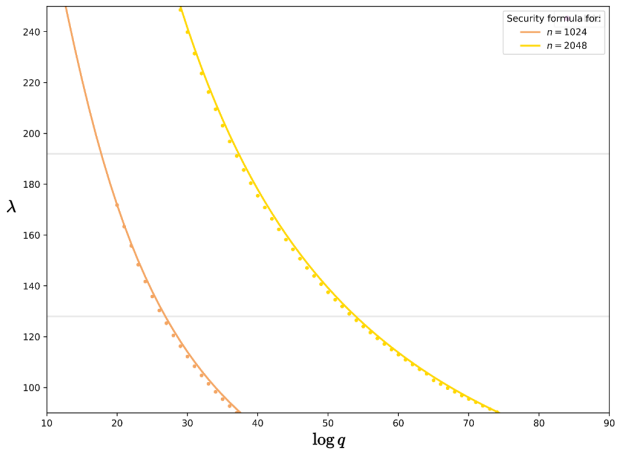

Fig. 2: The security level formula (Equation (24)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_2$ considering the BDD attack.

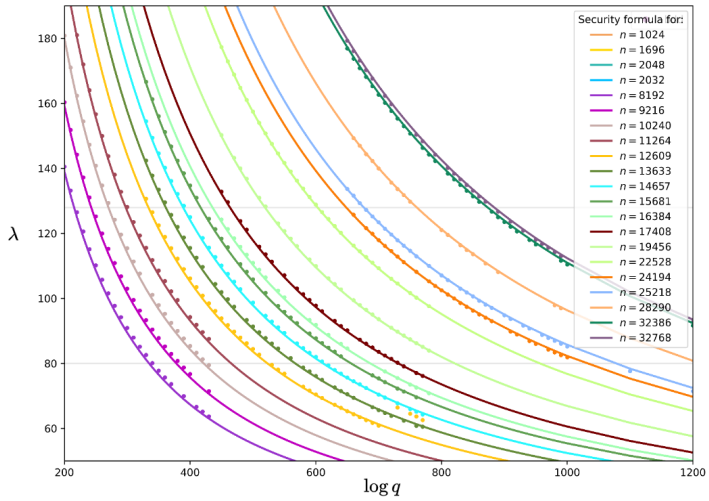

Fig. 3: The security formula (Equation (24)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$ considering the BDD attack.

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.

Our goal is to express Equation (24) 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 (12),

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

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 (12) 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 [HH08]. 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 (12) becomes

$$\beta \approx \frac{k_1 n \cdot \left(\ln\left(\frac{n}{\ln q}\right) + k\right)}{\left(k_4\sqrt{\ln q} + k_2\sqrt{n}\right)^2} \approx \frac{kn \cdot \left(\ln\left(\frac{n}{\ln q}\right) + k\right)}{\tilde{k}_4 \ln q + \tilde{k}_2 n + \tilde{k}_3\sqrt{n \ln q}}$$
$$\approx k_5 \frac{n}{\ln q} \ln\left(\frac{n}{\ln q} + k\right) + k_6, \tag{25}$$

where k_i are some constants.

Finally, substituting Equation (25) in Equation (24), we have

$$\lambda \approx a \frac{n}{\ln q} \ln\left(\frac{n}{\ln q} + k\right) + B \ln\left(2n^2 + d\right) + c, \tag{26}$$

where a, c, and d are constants. Note that Equation (26) is similar to Equation (23), and this is not surprising as the two attacks yield very similar results. Therefore, we aim to further approximate Equation (26) to obtain a formula identical to Equation (23), but with different constants. Thus, using coupled optimization, we obtain

$$\lambda \approx A' \ln\left(\frac{B'n}{\ln q}\right) \frac{n}{\ln q} + C' \ln n + D' \tag{27}$$

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 (24) and (27)) are presented in Tables 4 and 5.

$n = 2^{10}$				$n = 2^{11}$			
$\log q$	Estimator	(24)	(27)	$\log q$	Estimator	(24)	(27)
20	173	166	175	37	191	185	190
24	142	138	144	46	150	147	150
25	136	132	138	50	137	135	137
26	130	127	132	53	129	127	129
27	125	122	127	54	126	125	127
28	120	117	122	57	119	118	120
30	112	109	113	62	109	108	110
33	101	99	102	67	100	100	101
37	90	88	90	74	98	91	91
42	79	77	79	84	79	80	81

Table 4: Comparison between the security level provided by our formulas (Equations (24) and (27)) and the Lattice Estimator with with $\chi_s = \mathcal{U}_2$.

$n = 2^{10}$				$n = 2^{15}$			
$\log q$	Estimator	(24)	(27)	$\log q$	Estimator	(24)	(27)
16	227	234	232	650	181	179	179
18	200	206	202	760	150	151	150
19	189	194	190	810	141	140	140
25	140	144	140	880	129	128	128
27	129	132	128	930	120	121	120
28	124	127	123	1000	111	112	112
30	115	117	114	1050	106	107	106
32	107	109	106	1200	92	93	92
43	78	79	78	1400	80	80	79
48	70	70	70	1500	74	74	74

$n = 2^{16}$				$n = 2^{17}$			
$\log q$	Estimator	(24)	(27)	$\log q$	Estimator	(24)	(27)
1776	128	128	127	3576	128	129	127
1229	192	192	191	2469	192	192	190
955	256	256	254	1918	256	256	253

Table 5: Comparison between the security level provided by our formulas (Equations (24) and (27)) and the Lattice Estimator with $\chi_s = \mathcal{U}_3$.

4.4 Verification of the LWE dimension via uSVP, Equation (8)

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

$$n = \frac{A\lambda\big(0.5\ln(\lambda/0.292) + \ln(q/(2\pi e\sigma_e)) + B\big)^2}{0.584\ln(q/\zeta)\ln(\lambda/(0.584\pi e) + C)}, \tag{28}$$

where

$$A = 1.02575 \quad B = 0.17241 \quad C = 34.84910 \quad \text{if } \chi_s = \mathcal{U}_2$$
$$A = 1.05153 \quad B = 0.52652 \quad C = 43.20997 \quad \text{if } \chi_s = \mathcal{U}_3.$$

In Figure 4 we pictured the data points of the Lattice Estimator and our formula proposed in Equation (28) for the ternary distribution.

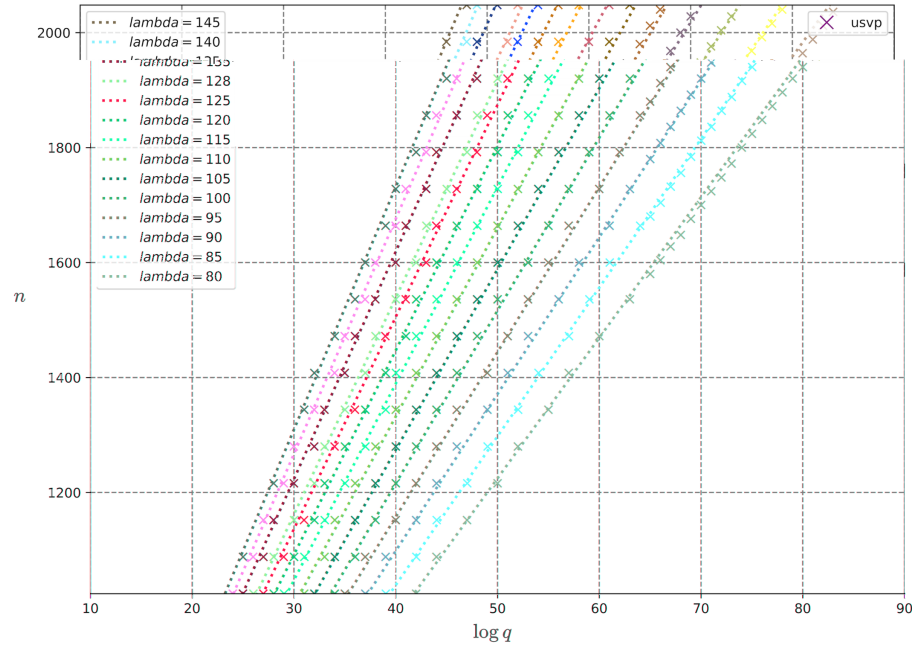

Fig. 4: The security level formula (Equation (28)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$ considering the uSVP attack.

Our goal is to express Equation (28) in a simplified form that explicitly depends on the variables λ and q. To do this, we consider Equation (23) 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 (23) approximating $x \approx a\lambda + b$, where a, b are some constants. Thus, using couple optimization, we obtain

$$n \approx \left(\frac{\lambda + A' \ln(\ln q)}{B' \ln(\lambda) + C'} + D' \right) \ln q, \tag{29}$$

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

The comparison results between the output of the Lattice Estimator and our formulas (Equation (29) and Equation (29)) are presented in Tables 6 and 7, demonstrating the effectiveness of our approach in accurately estimating security levels.

log q	$\text{Est}_{(28)}$	(28)	$\text{Est}_{(29)}$	(29)	log q	$\text{Est}_{(28)}$	(28)	$\text{Est}_{(29)}$	(29)
		$\lambda \approx 80$					$\lambda \approx 100$		
42	81	1036	81	1040	34	101	1037	101	1041
58	81	1432	81	1428	46	101	1402	101	1403
71	81	1754	80	1743	57	100	1736	100	1734
84	81	2076	80	2057	67	100	2039	100	2035
log q	$\text{Est}_{(28)}$	(28)	$\text{Est}_{(29)}$	(29)	log q	$\text{Est}_{(28)}$	(28)	$\text{Est}_{(29)}$	(29)
		$\lambda \approx 110$					$\lambda \approx 120$		
31	111	1037	112	1039	28	122	1019	122	1018
42	111	1400	111	1403	39	121	1412	121	1414
52	111	1731	111	1732	48	121	1734	121	1736
61	111	2029	111	2029	57	120	2056	121	2058
log q	$\text{Est}_{(28)}$	(28)	$\text{Est}_{(29)}$	(29)	log q	$\text{Est}_{(28)}$	(28)	$\text{Est}_{(29)}$	(29)
		$\lambda \approx 128$					$\lambda \approx 140$		
27	130	1045	130	1043	24	143	1015	142	1008
37	129	1424	129	1425	34	142	1425	142	1424
45	129	1727	129	1730	41	141	1713	142	1715
54	129	2069	129	2072	49	141	2041	141	2046

Table 6: Results of running the Lattice Estimator with the LWE dimension provided by Equations (28) and (29). Column $\text{Est}_{(28)}$ (resp. $\text{Est}_{(29)}$) shows the output of the Lattice Estimator run with parameters $\log q$, the output of Equation (28) (resp. (29)), secret distribution $\mathcal{U}_2$ and error distribution a discrete Gaussian with $\sigma_e = 3.19$.

log q	λ	$\text{Est}_{(28)}$	(28)	$\text{Est}_{(29)}$	(29)	log q	λ	$\text{Est}_{(28)}$	(28)	$\text{Est}_{(29)}$	(29)
43	80	82	1054	84	1082	1400	80	83	34247	81	33535
34	100	103	1036	105	1048	1100	100	102	33114	100	32607
32	110	114	1070	115	1075	1000	110	112	32873	110	32425
29	120	125	1057	125	1053	930	120	122	33117	120	32708
27	128	135	1050	133	1039	880	128	130	33244	128	32864
25	140	148	1063	145	1043	810	140	142	33207	140	32870

Table 7: Results of running the Lattice Estimator with the LWE dimension provided by Equations (28) and (29). Column λ shows the target security level provided as input to Equations (28) and (29). Column $\text{Est}_{(28)}$ (resp. $\text{Est}_{(29)}$) shows the output of the Lattice Estimator run with parameters $\log q$, the output of Equation (28) (resp. (29)), secret distribution $\mathcal{U}_3$ and error distribution a discrete Gaussian with $\sigma_e = 3.19$.

In Figure 5, we plot the data points of the Lattice Estimator for uSVP attack and our formula proposed in Equation (29).

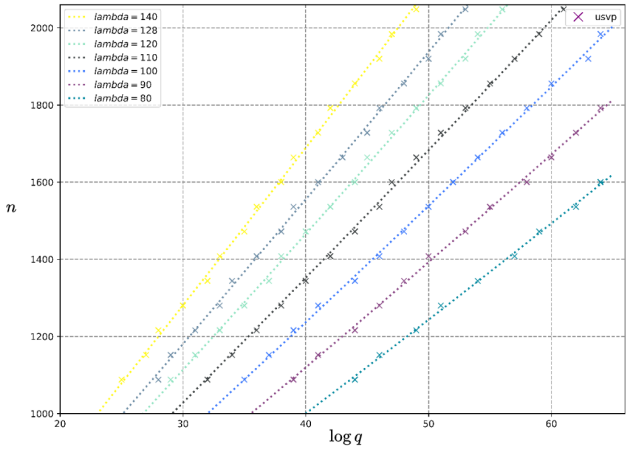

Fig. 5: Comparison between Equation (29) and the data points output by the Lattice Estimator for $\chi_s = \mathcal{U}_2$, considering the uSVP attack.

4.5 Verification of the LWE dimension via BDD, Equation (14)

Starting from Equation (14), 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):

$$n = \frac{(\tilde{A}\beta + \tilde{B})\left(2\ln q + \ln(\beta/2\pi e) + \tilde{C}\right)^2}{2\left(\ln(\beta/2\pi e) + \tilde{D}\right)\left(\ln(q^2/\zeta)\right)^2}\ln q \tag{30}$$

where

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

We approximate Equation (30) obtaining

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

for some constant $k_i \in \mathbb{R}$.

Using coupled optimization techniques we have

$$n = (A'\lambda + B' \ln \lambda)\left(C'\frac{\ln q}{\ln \lambda} + D'\frac{\ln \lambda}{\ln q}\right) \tag{31}$$

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

The comparison results between Equations (30) and (31) 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 (31).

log q	Est_n	(30)	Est_n	(31)	log q	Est_n	(30)	Est_n	(31)
		$\lambda \approx 80$					$\lambda \approx 100$		
42	81	1048	81	1050	34	101	1054	101	1055
58	81	1445	81	1445	46	101	1419	101	1419
71	81	1768	80	1766	57	101	1755	101	1754
84	81	2090	81	2087	67	101	2060	101	2059
log q	Est_n	(30)	Est_n	(31)	log q	Est_n	(30)	Est_n	(31)
		$\lambda \approx 110$					$\lambda \approx 120$		
31	111	1053	112	1055	28	122	1036	122	1037
42	111	1419	111	1418	39	121	1430	121	1430
52	111	1751	111	1751	48	121	1753	121	1753
61	111	2050	111	2050	57	121	2076	121	2078
log q	Est_n	(30)	Est_n	(31)	log q	Est_n	(30)	Est_n	(31)
		$\lambda \approx 128$					$\lambda \approx 140$		
27	130	1062	130	1063	24	143	1032	143	1033
37	129	1442	129	1442	34	142	1443	142	1442
45	129	1746	129	1746	41	141	1731	141	1730
54	129	2088	129	2090	49	141	2059	141	2061

Table 8: Results of running the Lattice Estimator with the LWE dimension provided by Equations (30) and (31). Column $\text{Est}_{(30)}$ (resp. $\text{Est}_{(31)}$) shows the output of the Lattice Estimator run with parameters $\log q$, the output of Equation (30) (resp. (31)), secret distribution $\mathcal{U}_2$ and error distribution a discrete Gaussian with $\sigma_e = 3.19$.

log q	λ	$\text{Est}_{(30)}$	(30)	$\text{Est}_{(31)}$	(31)	log q	λ	$\text{Est}_{(30)}$	(30)	$\text{Est}_{(31)}$	(31)
43	80	82	1076	80	1043	1400	80	82	33637	81	33675
34	100	106	1076	101	1028	1100	100	101	32761	101	32780
32	110	117	1117	111	1061	1000	110	111	32582	111	32604
29	120	130	1110	122	1047	930	120	121	32863	121	32892
27	128	140	1108	131	1039	880	128	129	33012	129	33047
25	140	156	1128	144	1051	810	140	142	32999	141	33043

Table 9: Results of running the Lattice Estimator with the LWE dimension provided by Equations (30) and (31). Column λ shows the target security level provided as input to Equations (30) and (31). Column $\text{Est}_{(30)}$ (resp. $\text{Est}_{(31)}$) shows the output of the Lattice Estimator run with parameters $\log q$, the output of Equation (30) (resp. (31)), secret distribution $\mathcal{U}_3$ and error distribution a discrete Gaussian with $\sigma_e = 3.19$.

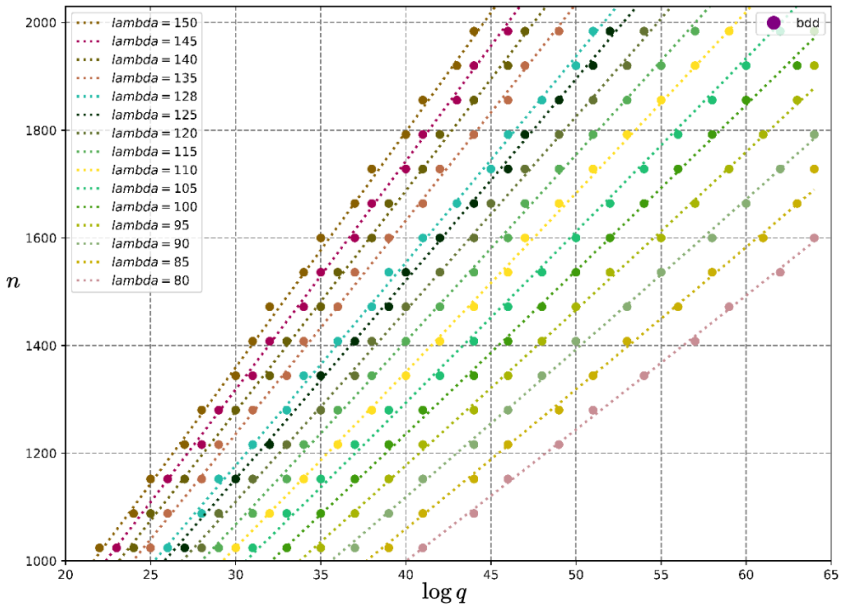

Fig. 6: The LWE dimension n (Equation (31)) with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$ for the BDD attack.

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 attack (uSVP, BDD, and hybrid). The idea is to first approximate the non-leading terms of these equations to retrieve the desired parameter as done in 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 two parameters on the $\log_2$-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.

uSVP attack. In Section 3.1, we obtained Equations (5) and (7), 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

$$\begin{cases} \beta - \frac{2n\ln(q/\zeta)\ln(\beta/(2\pi e))}{\ln^2(q\sqrt{\beta}/(2\pi e\sigma_e))} = 0 \\ \lambda - \left(0.292\beta + \log_2\left(8\sqrt{\frac{2n\ln(q/\zeta)\beta}{\ln(\beta/(2\pi e))}}\right) + 16.4\right) = 0. \end{cases} \tag{32}$$

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 (either λ, n, $\ln q$, or $\ln \sigma_e$) 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.

$n = 2^{10}$			
χ_s	$\log q$	Est_{num}	λ num
$\mathcal{U}_2$	13	266	268
	18	194	195
	27	127	126
	32	106	105
	42	80	78
	64	56	49

$n = 2^{11}$			
χ_s	$\log q$	Est_{num}	λ num
$\mathcal{U}_2$	28	261	260
	32	226	225
	37	193	192
	53	130	129
	64	106	105
	84	80	78

$n = 2^{10}$			
χ_s	$\log q$	Est_{num}	λ num
$\mathcal{U}_3$	14	265	263
	19	192	191
	27	131	130
	34	102	100
	43	79	78

$n = 2^{15}$			
χ_s	$\log q$	Est_{num}	λ num
$\mathcal{U}_3$	475	256	255
	611	192	190
	880	128	126
	1050	106	104
	1400	79	77

Table 10: Comparison between the security level provided by our numerical solver (num) and the Lattice Estimator for uSVP attack, when $\sigma_e = 3.19$.

Additionally, our tools provides numerical estimates for the LWE dimension n and the modulus q and standard deviation of the error distribution σ_e in bit size. The corresponding results, including the comparison with the Lattice Estimator output, are presented in Tables 11 to 13.

log q	λ	Est_{num}	n num
43	80	79	1047
34	100	98	1015
32	110	108	1042
29	120	118	1021
27	128	126	1008
25	140	137	1012

log q	λ	Est_{num}	n num
1400	80	80	33754
1100	100	100	32875
1000	110	110	32681
930	120	120	32935
880	128	129	33058
810	140	140	32999

Table 11: LWE dimension (n num) computed by the numerical solver given λ, $\log q$, $\chi_s = \mathcal{U}_3$ and $\chi_e = \mathcal{DG}(0, \sigma^2)$ with $\sigma_e = 3.19$. Column Est_{num} reports the security level from the Lattice Estimator using the output n *num* and the corresponding parameters, considering the uSVP attack.

χ_s	n	λ	Est_{num}	$\log q$ num
$\mathcal{U}_2$	1024	100	103	33
	1024	128	132	26
	1024	192	194	18
	1024	256	266	13
	2048	100	101	67
	2048	128	130	53
	2048	192	193	37
	2048	256	261	28

χ_s	n	λ	Est_{num}	$\log q$ num
$\mathcal{U}_3$	1024	100	102	34
	1024	128	131	27
	1024	192	204	19
	1024	256	265	14
	32768	100	101	1113
	32768	128	129	881
	32768	192	193	611
	32768	256	257	475

Table 12: Maximum $\log q$ values ensuring the target security level λ against the uSVP attack. Est_{num} reports the actual security levels obtained when plugging $(n, \log q \text{ num }, \chi_s, \sigma_e = 3.19)$ into the Lattice Estimator.

It is worth mentioning that, unlike the other parameters, the output of our numerical solver for σ_e can occasionally be off or fail to converge to a solution. Although this occurs very rarely (as shown in Table 13), we recommend that users perform an exhaustive search for optimal parameters using an LWE estimator. Our tool provides this option: specifically, it uses the result from the numerical solver as a starting point and performs a search over σ_e to find the optimal value according to the Lattice Estimator. The outcome of this approach is presented in the last three columns of Table 13. In particular, the column * Est_{num} provides the security levels obtained when plugging $(n, \log q, \chi_s = \mathcal{U}_2, *\ \log_2(\sigma_e))$ into the Lattice Estimator. Finally, the last column, $\text{Est}_{\text{calls}}$ shows the number of calls required for this correction.

n	$\log q$	λ	Est_{num}	$\log_2(\sigma_e)$	* Est_{num}	* $\log_2(\sigma_e)$	$\text{Est}_{\text{calls}}$
1024	32	100	102	0.33	100	-0.37	9
1024	64	100	100	31.91	100	31.61	5
1024	32	128	128	6.93	128	6.73	4
1024	64	128	128	38.83	128	38.73	3
1024	32	192	192	15.46	192	15.46	2
1024	64	192	191	47.41	192	47.51	2
1024	32	256	255	20.06	257	20.16	2
1024	64	256	255	52.03	257	52.13	2
2048	32	100	0	1.67	100	-14.09	2
2048	64	100	101	-1.11	100	-1.51	6
2048	32	128	129	-9.01	128	-9.11	3
2048	64	128	129	12.21	128	11.81	6
2048	32	192	193	-2.08	193	-2.08	2
2048	64	192	192	28.6	192	28.5	3
2048	32	256	256	5.26	256	5.26	2
2048	64	256	256	37.2	256	37.2	2

Table 13: Minimum standard deviation σ_e (in base-2 logarithm) of the error distribution for uSVP attacks with $\chi_s = \mathcal{U}_2$ obtained by running our numerical method approach. Column Est_{num} represents the security level computed by the Lattice Estimator for the corresponding $(n, \log q, \sigma_e)$ values. Columns * Est_{num} and * $\log_2(\sigma_e)$ show the results obtained by correcting the output in Est_{num} using the Lattice Estimator (column $\text{Est}_{\text{calls}}$ shows the number of calls required for the correction). A value of 0 in the column Est_{num} indicates that the method did not converge.

We want to emphasize that 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.

BDD 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,

$$\begin{cases} \beta - \dfrac{2n \ln q \ln\left(\frac{\beta}{2\pi e}\right)}{\left(\ln\left(\frac{\beta}{2\pi e}\right) + 2\ln\left(\frac{q}{\sigma_e\sqrt{2\pi e}}\right) - 2\sqrt{\frac{n}{2\ln q}\cdot\frac{\ln\left(\frac{\beta}{2\pi e}\right)}{\beta}}\ln\left(\frac{q}{\zeta}\right)\right)^2} = 0 \\ \lambda - (0.292\beta + \log_2(8d) + 16.4) = 0, \end{cases} \tag{33}$$

where the optimal d is set to $d = \sqrt{\frac{2n \ln q \ \beta}{\ln(\beta/2\pi e)}}$, and find the desired parameter by solving the system with a numerical method.

The comparison results between the output of the Lattice Estimator and our numerical method are presented in Table 14.

	$n = 2^{10}$				$n = 2^{11}$		
χ_s	$\log q$	Est_{num}	λ num	χ_s	$\log q$	Est_{num}	λ num
$\mathcal{U}_2$	13	264	272	$\mathcal{U}_2$	28	258	261
	18	191	196		32	224	226
	27	125	127		37	191	192
	32	105	105		53	129	129
	42	79	79		64	105	105
	64	58	49		84	79	78
	$n = 2^{10}$				$n = 2^{15}$		
χ_s	$\log q$	Est_{num}	λ num	χ_s	$\log q$	Est_{num}	λ num
$\mathcal{U}_3$	14	261	265	$\mathcal{U}_3$	475	256	255
	19	189	192		611	192	190
	27	129	130		880	129	126
	34	100	101		1050	106	104
	43	78	78		1400	80	77

Table 14: Comparison between the security level provided by our numerical solver (num) and the Lattice Estimator for BDD attack, when $\sigma_e = 3.19$.

In the computation of n, $\log q$ 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 $\log q$ or $\log \sigma_e$. Therefore, in this case we use the version of Equation (11) in which η is not replaced 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.

The results produced by our numerical solver, along with the comparison with the Lattice Estimator output for n, q, and σ_e, are presented in Tables 15 to 17, respectively.

log q	λ	Est_{num}	n num
43	80	79	1031
34	100	98	1005
32	110	108	1034
29	120	118	1014
27	128	126	1001
25	140	137	1006

log q	λ	Est_{num}	n num
1400	80	80	32955
1100	100	100	32374
1000	110	110	32260
930	120	120	32569
880	128	129	32726
810	140	140	32712

Table 15: LWE dimension (n num) computed by the numerical solver given λ, $\log q$, $\chi_s = \mathcal{U}_3$ and $\chi_e = \mathcal{DG}(0, \sigma^2)$ with $\sigma_e = 3.19$. Column Est_{num} reports the security level from the Lattice Estimator using the output n *num* and the corresponding parameters, considering the BDD attack.

As before, for σ_e, we recommend that users perform an exhaustive search for optimal parameters using our tool. The outcome of this process is reported in the last three columns of Table 17. In particular, the column Est_{num} shows the security levels obtained by plugging $(n, \log q, \chi_s = \mathcal{U}_2, *\log_2(\sigma_e))$ into the Lattice Estimator; and the last column ($\text{Est}_{\text{calls}}$) indicates the number of calls needed to complete this refinement.

χ_s	n	λ	Est_{num}	$\log q$ num
	1024	100	101	33
	1024	128	130	26
	1024	192	191	18
$\mathcal{U}_2$	1024	256	264	13
	2048	100	100	67
	2048	128	129	53
	2048	192	191	37
	2048	256	258	28

χ_s	n	λ	Est_{num}	$\log q$ num
	1024	100	100	34
	1024	128	129	27
	1024	192	189	18
$\mathcal{U}_3$	1024	256	261	14
	32768	100	100	1096
	32768	128	129	872
	32768	192	192	608
	32768	256	256	474

Table 16: Maximum $\log q$ values ensuring the target security level λ against the BDD attack. Est_{num} reports the actual security levels obtained when plugging $(n, \log q \text{ num }, \chi_s, \sigma_e = 3.19)$ into the Lattice Estimator.

Hybrid attack. In Section 3.3, we derived three equations that relate the hybrid BDD attack parameters:

$$\begin{cases} 0.292\beta + \log_2(8d) + 16.4 = \log_2\left(\binom{n_g}{\omega} \cdot 2^{\omega}\right), \\ d - n_g = \left\lceil \sqrt{\frac{n \ln q}{\ln \delta_\beta}} \right\rceil, \\ (-d+1)\log_2(\delta_\beta) + \frac{1}{d}\left((d - n + n_g - 1) \cdot \log_2 q + (n - n_g)\log_2(\zeta)\right) \geq 2\log(\sigma_e^2). \end{cases}$$

n	$\log q$	λ	Est_{num}	$\log_2(\sigma_e)$	* Est_{num}	* $\log_2(\sigma_e)$	$\text{Est}_{\text{calls}}$
1024	32	100	98	-0.19	100	0.21	5
1024	64	100	98	28.42	100	28.72	4
1024	32	128	126	6.38	128	6.78	5
1024	64	128	164	34.25	132	31.95	25
1024	32	192	188	13.64	192	13.94	4
1024	64	192	293	41.33	193	36.03	55
1024	32	256	301	17.25	276	16.35	11
1024	64	256	389	45.13	257	39.63	57
2048	32	100	99	-14.19	100	-13.99	3
2048	64	100	99	-1.45	100	-1.25	3
2048	32	128	126	-9.13	128	-8.93	3
2048	64	128	127	11.28	128	11.58	4
2048	32	192	190	-2.12	192	-1.92	3
2048	64	192	190	25.71	192	25.91	3
2048	32	256	253	4.92	256	5.22	4
2048	64	256	254	32.82	256	33.02	3

Table 17: Minimum standard deviation σ_e (in base-2 logarithm) of the error distribution for BDD attacks with $\chi_s = \mathcal{U}_2$ obtained by running our numerical method approach. Column Est_{num} represents the security level computed by the Lattice Estimator for the corresponding $(n, \log q, \sigma_e)$ values. Columns * Est_{num} and * $\log_2(\sigma_e)$ show the results obtained by correcting the output in Est_{num} using the Lattice Estimator (column $\text{Est}_{\text{calls}}$ shows the number of calls required for the correction).

Even though this system has four unknowns (β, d, n_g, ω) for the given LWE parameters n, q, σ_e, h, we can brute-force over ω as the optimal values do not exceed 40 for the interesting LWE parameters. To aid numerical solvers, we approximate the binomial coefficient from the first equation using Sterling's formula, i.e., $\log_2 \binom{n_g}{\omega} \approx n_g \mathcal{H}(\omega/n_g)$, where $\mathcal{H}()$ is the binary entropy function.

Having found β, d, n_g, ω for the given LWE parameters, we can easily compute the security level λ using Equation (13). The results are shown in Table 18.

χ_s	n	log q	h	Est_{num}	λ num
		200	128	128	127
		119	128	195	196
$\mathcal{HWT}(h)$	8192	87	128	243	239
		210	192	128	127
		128	192	203	205
		91	192	280	274

χ_s	n	log q	h	Est_{num}	λ num
		850	128	125	122
		500	128	193	190
$\mathcal{HWT}(h)$	32768	330	128	255	254
		850	192	128	126
		565	192	191	188
		410	192	249	248

Table 18: Comparison between the security level provided by our numerical solver (λ num) and the Lattice Estimator for the hybrid attack for sparse ternary secrets of Hamming weight h.

Oppositely, in order to find the optimal modulus q given the desired security level λ, we build the system

$$\begin{cases} \lambda = 0.292\beta + \log_2(8d) + 16.4 - \log_2(p_{\text{succ}}(n_g, w)), \\ \lambda = \log_2\left(\binom{n_g}{\omega} \cdot 2^{\omega}\right) - \log_2(p_{\text{succ}}(n_g, w)) \\ d - n_g = \left\lceil \sqrt{\frac{n \ln q}{\ln \delta_\beta}} \right\rceil, \\ (-d+1)\log_2(\delta_\beta) + \frac{1}{d}\left((d - n + n_g - 1) \cdot \log_2 q + (n - n_g)\log_2(\zeta)\right) \geq 2\log(\sigma_e^2), \end{cases}$$

where $p_{\text{succ}}(n_g, w)$ is given in Equation (16).

In Table 19, we show the output of our numerical solver for the system above, providing the largest $\log q$ ensuring the target λ.

χ_s	n	h	λ	Est_{num}	$\log q$ num
		128	100	102	34
		192	100	100	36
		128	128	132	26
$\mathcal{HWT}(h)$	1024	192	128	131	28
		128	192	194	17
		192	192	197	19
		128	256	238	14
		192	256	249	15

χ_s	n	h	λ	Est_{num}	$\log q$ num
		128	80	81	1360
		192	80	79	1403
		128	128	121	877
$\mathcal{HWT}(h)$	32768	192	128	125	877
		128	192	163	629
		192	192	178	608
		128	256	218	420
		192	256	211	500

Table 19: Maximum $\log q$ values ensuring the target security level λ against the hybrid attack, for sparse ternary secrets of Hamming weight h. Est reports the actual security levels obtained when plugging $(n, \log q \text{ num}, h)$ into the Lattice Estimator.

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[10], 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 rely on the Lattice Estimator or on the Leaky-LWE Estimator [DSDGR20].[11] 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 basic usage overview of our tool's functionality, we refer the reader to our Github repository for more examples and a detailed explanation of all the functionalities of our tool.

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:

```
python3 src/estimate.py --param "lambda" --n "1024" --logq
    "20;35;40" --secret "binary" --std "3.19"
```

gives as output the following:

[10] https://github.com/Crypto-TII/fastparameterselection
[11] https://github.com/lducas/leaky-LWE-Estimator

```
secret dist.   | lwe dim. | log q | output
---------------+----------+-------+-------
Uniform (-1 0) | 1024     | 20    | 173
Uniform (-1 0) | 1024     | 35    | 95
Uniform (-1 0) | 1024     | 40    | 83
```

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 together with target λ. For instance:

```
python3 src/estimate.py --param "n" --lambda "128" --logq
    "27;37;45;54" --secret "binary" --error "gaussian" --std
    "3.19"
```

```
secret dist.   | lambda | log q | output | pow
---------------+--------+-------+--------+-----
Uniform (-1 0) | 128    | 27    | 1063   | 1024
Uniform (-1 0) | 128    | 37    | 1442   | 1024
Uniform (-1 0) | 128    | 45    | 1746   | 2048
Uniform (-1 0) | 128    | 54    | 2090   | 2048
```

The last column `pow` shows the closest to the `output` power-of-two.

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:

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

```
secret dist.   | lambda | lwe dim. | output
---------------+--------+----------+-------
Uniform (-1 1) | 128    | 32768    | 881
```

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"`:

```
python3 src/estimate.py --param "std_e" --lambda "192" --n
    "2048" --logq "64" --secret "binary"
```

```
secret dist.   | lambda | lwe dim. | log q | output
---------------+--------+----------+-------+-------
Uniform (-1 0) | 192    | 2048     | 64    | 28.60
```

6.1 Adapting Our Tool to Non-FHE Settings

While our formulas are fine-tune for FHE settings, our tool allows users to input arbitrary LWE parameters $(\lambda, n, \sigma_e, \sigma_s, q)$. In particular, σ_s and σ_e can be chosen from the following distributions:

- Uniform binary distribution $\mathcal{U}_2$
- Uniform ternary distribution $\mathcal{U}_3$
- Uniform modulus distribution $\mathcal{U}_p$
- Uniform distribution $\mathcal{U}_{[a,b]}$
- Sparse ternary distribution $\mathcal{HWT}(h)$
- Discrete Gaussian distribution $\mathcal{DG}(0, \sigma^2)$
- The centered binomial distribution ψ_η.

Some cases in which this flexibility is necessary are, for example, Kyber and Saber. We refer the reader to our Github repository for examples.

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 [APS15] 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 [BCC+24] and [BBB+23] which, at the time of writing, are the state-of-the-art approaches for parameter selection in FHE.

7.1 Comparision with [BCC+24]

In [BCC+24], 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.[12] 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 [BCC+24] is the scope of parameters that an end-user can obtain. That is, a table-based approach such as the one provided by [BCC+24] 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 [BCC+24] 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 [DGM24]. With our tool, we can quickly get the parameters of Table 20, which would serve the purpose of such applications.

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 20: 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.

Another difference between our work and [BCC+24] is their use of the Lattice Estimator. The script provided by [BCC+24] 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 [BCC+24] relies on it to produce the tables.

There are other subtle but important differences between [BCC+24] and our work. They use the Matzov attack [MAT22], which is not known to be correct [DP23], we are relying on more understood attacks. Properly analyzing dual attacks on lattice in the same fashion as we present here for primal attacks (uSVP, BDD, and hybrid) is beyond the scope of this work. Finally, they do not consider sparse secrets nor NTRU while we do not consider quantum attacks.

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

7.2 Comparison with [BBB+23]

In [BBB+23], 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[13], 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 [BBB+23]. 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, (bit size of) modulus q, standard deviations of secret σ_s, and error σ_e. By 'reversing' these formulas we can express any fixed LWE parameter n, or $\log q$, or σ_s, or $\log \sigma_e$ 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 [APS15]. 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 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.

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

Bibliography

[AAUC18] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4):1–35, 2018. 2

[ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178. Springer, Berlin, Heidelberg, August 2016. 4

[ACC+18] Martin R Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, `HomomorphicEncryption.org`, Toronto, Canada, November 2018. 2, 9

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-Quantum Key Exchange: A New Hope. In *Proceedings of the 25th USENIX Conference on Security Symposium*, page 327–343, 2016. 8, 11

[AGVW17] Martin R Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving uSVP and applications to LWE. In *Advances in Cryptology–ASIACRYPT 2017*, pages 297–322, 2017. 3, 10, 11

[AKP24] Andreea Alexandru, Andrey Kim, and Yuriy Polyakov. General functional bootstrapping using CKKS. *Cryptology ePrint Archive*, 2024. 17

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129. Springer, Cham, April / May 2017. 10, 16

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. 2, 4, 8, 12, 18, 38, 40

[Bab86] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. 8

[BBB+22] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff,

Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. OpenFHE: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915, 2022. 5

[BBB+23] Loris Bergerat, Anas Boudi, Quentin Bourgerie, Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Parameter Optimization and Larger Precision for (T)FHE. *Journal of Cryptology*, 36(3):28, 2023. 3, 4, 5, 38, 40

[BCC+24] Jean-Philippe Bossuat, Rosario Cammarota, Jung Hee Cheon, Ilaria Chillotti, Benjamin R. Curtis, Wei Dai, Huijing Gong, Erin Hales, Duhyeong Kim, Bryan Kumara, Changmin Lee, Xianhui Lu, Carsten Maple, Alberto Pedrouzo-Ulloa, Rachel Player, Luis Antonio Ruiz Lopez, Yongsoo Song, Donggeon Yhee, and Bahattin Yildiz. Security guidelines for implementing homomorphic encryption. *Cryptology ePrint Archive*, 2024. 2, 5, 38, 39

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *SODA 2016*, pages 10–24. SIAM, 2016. 8

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In *2018 IEEE EuroS&P*, pages 353–367, 2018. 8

[Ber23] Daniel J. Bernstein. Asymptotics of hybrid primal lattice attacks. Cryptology ePrint Archive, Report 2024/592, 2023. 16

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014. 2

[BMCM23] Beatrice Biasioli, Chiara Marcolla, Marco Calderini, and Johannes Mono. Improving and Automating BFV Parameters Selection: An Average-Case Approach. *Cryptology ePrint Archive, Paper 2023/600*, 2023. 3

[Bra12] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In *Advances in Cryptology – CRYPTO 2012*, pages 868–886, 2012. 2

[CDS15] Sergiu Carpov, Paul Dubrulle, and Renaud Sirdey. Armadillo: a compilation chain for privacy preserving applications. In *Proceedings of the 3rd International Workshop on Security in Cloud Computing*, pages 13–19, 2015. 3

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Berlin, Heidelberg, December 2016. 2, 9

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast Fully Homomorphic Encryption over the Torus. *Journal of Cryptology*, 33(1):34–91, 2020. 2, 9

[CH18] Hao Chen and Kyoohyung Han. Homomorphic lower digits removal and improved FHE bootstrapping. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 315–337, Cham, 2018. Springer International Publishing. 17

[CHK+18] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full RNS variant of approximate homomorphic encryption. In *International Conference on Selected Areas in Cryptography – SAC 2018*, pages 347–368. Springer, 2018. 2

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437. Springer, Cham, December 2017. 2, 16

[CMHSJP25] Kevin Carrier, Charles Meyer-Hilfiger, Yixin Shen, and Tillich Jean-Pierre. Assessing the impact of a variant of the latest dual attack, 2025. https://eprint.iacr.org/2022/1750. 3

[Con23] HEIR Contributors. HEIR: Homomorphic Encryption Intermediate Representation, 2023. https://github.com/google/heir. 3

[CPS18] Eric Crockett, Chris Peikert, and Chad Sharp. Alchemy: A language and compiler for homomorphic encryption made easy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1020–1037, 2018. 3

[CS16] Anamaria Costache and Nigel P Smart. Which ring based somewhat homomorphic encryption scheme is best? In *Cryptographers' Track at the RSA Conference*, pages 325–340. Springer, 2016. 19

[CS17] Anamaria Costache and Nigel P. Smart. Homomorphic encryption without gaussian noise. Cryptology ePrint Archive, Paper 2017/163, 2017. 19

[DGM24] Andrea Di Giusto and Chiara Marcolla. Breaking the power-of-two barrier: noise estimation for BGV in NTT-friendly rings. *Designs, Codes and Cryptography*, Nov 2024. 39

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, Feb. 2018. 8

[DKS+20] Roshan Dathathri, Blagovesta Kostova, Olli Saarikivi, Wei Dai, Kim Laine, and Madan Musuvathi. EVA: An encrypted vector arithmetic language and compiler for efficient homomorphic computation. In *Proceedings of the 41st ACM SIGPLAN Con-*

ference on Programming Language Design and Implementation, pages 546–561, 2020. 3

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 617–640, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. 2

[DP23] Léo Ducas and Ludo N. Pulles. Does the dual-sieve attack on learning with errors even work? In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 37–69. Springer, Cham, August 2023. 3, 9, 10, 39

[DSDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with Side Information: Attacks and Concrete Security Estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 329–358. Springer International Publishing, 2020. 8, 36

[DvW21] Léo Ducas and Wessel P. J. van Woerden. NTRU fatigue: How stretched is overstretched? In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 3–32. Springer, Cham, December 2021. 4

[FHK+18] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhan. FALCON: Fast-fourier lattice-based compact signatures over NTRU. *Submission to the NIST's post-quantum cryptography standardization process*, 36(5):1–75, 2018. 8

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012. 2

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*, volume 20. Stanford university Stanford, 2009. 2

[GJ21] Qian Guo and Thomas Johansson. Faster dual lattice attacks for solving LWE with applications to CRYSTALS. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 33–62. Springer, Cham, December 2021. 3, 9

[GV23] Robin Geelen and Frederik Vercauteren. Bootstrapping for BGV and BFV Revisited. *Journal of Cryptology*, 36(2):12, 2023. 17

[HG07] Nick Howgrave-Graham. A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. In *Advances in Cryptology - CRYPTO 2007*, pages 150–169, 2007. 3, 10

[HH08] Abdolhossein Hoorfar and Mehdi Hassani. Inequalities on the lambert w function and hyperpower function. *J. Inequal. Pure and Appl. Math*, 9(2):5–9, 2008. 24

[HKM18] Gottfried Herold, Elena Kirshanova, and Alexander May. On the asymptotic complexity of solving lwe. *Des. Codes Cryptography*, 86(1):55–83, jan 2018. 9

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In *Advances in Cryptology – CRYPTO 2011*, pages 447–464, 2011. 8

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 193–206, 1983. 3, 10

[KF17] Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched NTRU parameters. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 3–26. Springer, Cham, April / May 2017. 4

[KMR24] Elena Kirshanova, Chiara Marcolla, and Sergi Rovira. Guidance for efficient selection of secure parameters for fully homomorphic encryption. In *International Conference on Cryptology in Africa*, pages 376–400. Springer, 2024. 2, 3

[Lat] Lattigo. http://github.com/ldsec/lattigo. 5

[LN13] Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *LNCS*, pages 293–309. Springer, Berlin, Heidelberg, February / March 2013. 3, 12, 13, 16

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 1–23, 2010. 2, 9

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, jun 2015. 9

[LW24] Zeyu Liu and Yunhao Wang. Relaxed functional bootstrapping: A new perspective on bgv/bfv bootstrapping. *Cryptology ePrint Archive*, 2024. 17

[MAT22] MATZOV. Report on the security of LWE: Improved dual lattice attack, April 2022. https://zenodo.org/records/6412487. 3, 9, 39

[MHWW24] Shihe Ma, Tairong Huang, Anyu Wang, and Xiaoyun Wang. Accelerating BGV bootstrapping for large p using null polynomials over $\mathbb{Z}_{p^e}$. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 403–432. Springer, Cham, May 2024. 19

[MML+23] Johannes Mono, Chiara Marcolla, Georg Land, Tim Güneysu, and Najwa Aaraj. Finding and evaluating parameters for bgv. In *International Conference on Cryptology in Africa – AFRICACRYPT 2023*, pages 370–394. Springer, 2023. 2, 3, 4

[MSM17] Paulo Martins, Leonel Sousa, and Artur Mariano. A survey on fully homomorphic encryption: An engineering perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–33, 2017. 2

[MSM+22] Chiara Marcolla, Victor Sucasas, Marc Manzano, Riccardo Bassoli, Frank HP Fitzek, and Najwa Aaraj. Survey on fully homomorphic encryption, theory, and applications. *Proceedings of the IEEE*, 110(10):1572–1609, 2022. 1, 2, 3

[Pai] Pascal Paillier. Invited talk: Recent advances in homomorphic compilation. https://youtu.be/phWYLwlPTY0?si=gwcf8svL6tOYcizv. 3

[PS24] Amaury Pouly and Yixin Shen. Provable dual attacks on learning with errors. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 256–285. Springer, Cham, May 2024. 3

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93, 2005. 2, 9

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2):201–224, 1987. 7, 11

[Sch03] Claus Peter Schnorr. Lattice reduction by random sampling and birthday methods. In Helmut Alt and Michel Habib, editors, *STACS 2003*, pages 145–156, 2003. 8

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, 1994. 11

[SEA19] Microsoft SEAL (release 3.4). https://github.com/Microsoft/SEAL, October 2019. Microsoft Research, Redmond, WA. 5

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 617–635. Springer, 2009. 9

[vEPIL19] Tim van Elsloo, Giorgio Patrini, and Hamish Ivey-Law. SEALion: A framework for neural network inference on encrypted data. *arXiv preprint arXiv:1904.12840*, 2019. 3

[VJH21] A. Viand, P. Jattke, and A. Hithnawi. SoK: Fully Homomorphic Encryption Compilers. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1092–1108. IEEE Computer Society, 2021. 3