Finding and Evaluating Parameters for BGV

Johannes Mono[1], Chiara Marcolla[2], Georg Land[1,3], Tim Güneysu[1,3], and Najwa Aaraj[2]

[1] Ruhr University Bochum, Horst Görtz Institute for IT Security, Germany
[2] Technology Innovation Institute, Abu Dhabi, United Arab Emirates
[3] DFKI GmbH, Cyber-Physical Systems, Bremen, Germany

Abstract. Fully Homomorphic Encryption (FHE) is a groundbreaking technology that allows for arbitrary computations to be performed on encrypted data. State-of-the-art schemes such as Brakerski Gentry Vaikuntanathan (BGV) are based on Learning with Errors over rings (RLWE) assumption and each ciphertext has an associated error that grows with each homomorphic operation. For correctness, the error needs to stay below a certain threshold, requiring a trade-off between security and error margin for computations in the parameters. Choosing the parameters accordingly, for example, the polynomial degree or the ciphertext modulus, is challenging and requires expert knowledge specific to each scheme.

In this work, we improve the parameter generation process across all steps of its process. We provide a comprehensive analysis for BGV in the Double Chinese Remainder Theorem (DCRT) representation providing more accurate and better bounds than previous work on the DCRT, and empirically derive a closed formula linking the security level, the polynomial degree, and the ciphertext modulus. Additionally, we introduce new circuit models and combine our theoretical work in an easy-to-use parameter generator for researchers and practitioners interested in using BGV for secure computation.

Our formula results in better security estimates than previous closed formulas while our DCRT analysis results in reduced prime sizes of up to 42 % compared to previous work.

Keywords: Fully Homomorphic Encryption, BGV Scheme, Parameter Generation, RLWE Security, DCRT Representation

1 Introduction

Since Gentry's seminal work in 2009 [14], fully homomorphic encryption (fully homomorphic encryption (FHE)) has attracted much attention from the cryptographic research community [1, 22, 23]. FHE enables arbitrary computations on encrypted data and thus opens up new possibilities in data processing. As an example, hospitals analyzing health information can work only on encrypted data and provide clients with an encrypted result, thus not risking leaking any sensitive data.

The BGV scheme [7] is currently considered one of the state-of-the-art FHE schemes. BGV is based on the Learning with Errors (LWE) problem and its ring variant RLWE [21, 26]. RLWE-based FHE schemes, including BGV, need to keep the error associated with each ciphertext below a certain threshold, as decryption would fail otherwise. This requires a trade-off between security (small ciphertext modulus), and error margin (implying a large ciphertext modulus) [22].

In general, choosing parameters for FHE schemes such as the polynomial degree d or the ciphertext modulus q is challenging and requires expert knowledge specific to each scheme. Multiple parameters must be considered, and users and developers alike need to take many deliberate choices when using state-of-the-art software libraries. A real-world example is the programming interface of PALISADE [25], an open-source FHE library that also implements BGV. A user needs to choose a polynomial implementation as well as seven additional parameters that all influence the polynomial degree as well as the ciphertext modulus (Listing 1.1).

```
auto ctx = CryptoContext<DCRTPoly>::BGVrns(
    2,                  // cyclic order
    65537,              // plaintext modulus
    HEStd_128_classic,  // security level
    3.2,                // error standard deviation
    2,                  // multiplicative depth
    OPTIMIZED,          // secret distribution
    BV);                // key switching method
```

Listing 1.1. BGV setup routine in PALISADE

This flexibility can be valuable for researchers familiar with FHE. For other users, however, this burden of choice increases the difficulty of generating a reasonable and secure FHE instance.

There are several challenges within the parameter selection process that need to be addressed in order to choose correct and secure parameters. Given a FHE scheme, we first adjust the error growth analysis depending on implementation choices as well as use-case-specific requirements and then compute error bounds for each individual operation. Using these bounds, we model the homomorphic circuit and determine a lower bound on the ciphertext modulus. Finally, we need to select the polynomial degree sufficiently large enough to provide a secure scheme instantiation for the desired security level. In the following, we will highlight the current state-of-the-art for each step in this process.

Related Work. For BGV, there currently are two main approaches to analyzing the error growth: the canonical norm [9, 10, 18] or the infinity norm [19]. While the canonical norm is known to result in better parameters, the only work analyzing BGV operations in the Double Chinese Remainder Theorem (DCRT) representation is the latter using the less optimal infinity norm. Since current state-of-the-art libraries all use the DCRT for efficient implementation, correctly modeling it in the bound computation is essential. Costache et al. [11] present an *average-case noise* analysis approach for BGV which is tailored to the dynamic

noise estimation in HElib [17]. However, while there are case studies with specific parameter sets [8], the more general circuit models for static parameter generation have remained simple, excluding for example operations such as rotations [15] or the constant multiplication [19].

The FHE community made a significant stride towards standardization by taking on the last step of the parameter generation process, culminating in developing the Homomorphic Encryption Standard [3]. The standard provides parameter sets based on the Lattice Estimator [5], a software tool to determine the security level of RLWE instances. More specifically, the standard provides upper limits on the size of the ciphertext modulus for certain security levels λ and polynomial degrees d in the form of lookup tables. Libraries such as PALISADE then use these lookup tables for fast security estimation during parameter generation. However, as FHE parameters depend on the specific use case, the trade-off is non-optimized parameter sets leading to larger parameters than necessary. This, in turn, negatively affects the runtime and memory usage of FHE implementations and – as a consequence – leads to significantly less performance.

Contribution The main idea of our work is to improve the current state-of-the-art of parameter selection process in all three steps of the process. Moreover, we aim to improve the usability of the BGV scheme. To achieve this goal, our work provides the following contributions:

- We provide a comprehensive analysis of noise growth in BGV using the canonical embedding norm for the DCRT representation, the current state-of-the-art for implementing RLWE-based schemes. To the best of our knowledge, these bounds are currently the best theoretical bounds for the BGV scheme with the DCRT representation.
- We empirically derive a closed formula computing the security for a given ciphertext modulus and polynomial degree. This enables the fast evaluation of a security estimate in the last step of the parameter selection process (Section 3.7).
- We provide new circuit models considering additional cases such as rotations or constant multiplications resulting in closer matching noise estimates for different use cases (Section 3).
- Using our theoretical and empirical formulas, we provide an interactive parameter generator for the leveled BGV scheme (Section 3.8). Additionally, the generator outputs easy-to-use code snippets for the generated parameters for multiple state-of-the-art libraries.

2 Preliminaries

2.1 Notations

We start with general notations we will use in the remainder of this work.

For a positive integer m, we denote by $\mathbb{Z}_m$ the ring of integers modulo m. We denote by $\mathbb{Z}_m^* = \{x \in \mathbb{Z}_m \,|\, (x, m) = 1\}$ the multiplicative group of units. We

denote by $R = \mathbb{Z}[x]/\langle \Phi_m(x) \rangle$ and by $R_p = \mathbb{Z}_p[x]/\langle \Phi_m(x) \rangle$, where p is an integer and $\Phi_m(x)$ is the cyclotomic polynomial (see Section 2.2). We denote by t and q the plaintext and the ciphertext modulus, respectively, and R_t the plaintext space. Moreover, we set $t \equiv 1 \mod m$ and q a chain of primes, such that

$$q = q_{L-1} = \prod_{j=0}^{L-1} p_j,$$

where p_i (for $1 \leq i \leq L-2$) are roughly of the same size and $p_j \equiv 1 \mod m$ [15]. The multiplicative depth M of the circuit determines the number of primes $L = M + 1$. Thus, for any level ℓ, we have $q_\ell = \prod_{j=0}^{\ell} p_j$.

Polynomials are denoted by lower letters such as a, vectors of polynomials are denoted in bold $\mathbf{a}$. Polynomial multiplication is denoted as $a \cdot b$ while multiplication with a scalar t is denoted as ta. Let $x \in \mathbb{R}$, we write $[x]_m \in [-m/2, m/2)$ for the centered representative of $x \bmod m$.

We denoted by χ_e the RLWE error distribution, typically a discrete Gaussian with standard deviation $\sigma = 3.19$ [3], and by χ_s the secret key distribution. In general, if χ is a probabilistic distribution and $a \in R$ a random polynomial, we write $a \leftarrow \chi$ when sampling each coefficient independently from χ.

2.2 Mathematical Background

Cyclotomic polynomial Let $\mathbb{F}$ be a field and m be a positive integer. We recall that a m-th *root of unity* is a number $\zeta \in \mathbb{F}$ satisfying the equation $\zeta^m = 1$. It is called *primitive* if m is the smallest positive integer for which $\zeta^m = 1$. The m-th *cyclotomic polynomial* is defined as $\Phi_m(x) = \prod_{(j,m)=1}(x - \zeta^j)$. The degree of Φ_m is $\phi(m) = m \prod_{p|m}(1 - 1/p) = |\mathbb{Z}_m^*|$, Euler's totient function.

Canonical embedding and norms In this section, we recall the result of [9, 10, 18].

Let $a \in R$ be a polynomial. We recall that the *infinity norm* of a is defined as $||a||_\infty = \max\{|a_i| : 0 \leq i \leq \phi(m) - 1\}$. If we consider two polynomials $a, b \in R$, the infinity norm of their product is bounded by $||ab||_\infty \leq \delta_R ||a||_\infty ||b||_\infty$, where δ_R is the *expansion factor*.

The *canonical embedding* of a is the vector obtained by evaluating a at all primitive m-th roots of unity. The *canonical embedding norm* of $a \in R$ is defined as $||a||^{can} = \max_{j \in \mathbb{Z}_m^*} |a(\zeta^j)|$. For a vector of polynomials $\mathbf{a} = (a_0, \ldots, a_{n-1}) \in R^n$, the canonical embedding norm is defined as $||\mathbf{a}||^{can} = \max_i ||a_i||^{can}$. For any polynomial $a, b \in R$, the following properties hold:

- $||a||^{can} \leq \phi(m)||a||_\infty$.
- $||ab||^{can} \leq ||a||^{can}||b||^{can}$.
- $||a||_\infty \leq c_m ||a||^{can}$ for the *ring expansion factor* c_m.

Note that, $c_m = 1$ if the degree of $\Phi_m(x)$ is a power-of-two [12].

Let us consider a random $a \in R$ where each coefficient is sampled independently from one of the following zero-mean distributions:

- $\mathcal{DG}_q(\sigma^2)$, the discrete Gaussian distribution with standard deviation σ over the interval $(-q/2, q/2]$.
- $\mathcal{DB}_q(\sigma^2)$, the discrete Binomial distribution with standard deviation σ over the interval $(-q/2, q/2]$.
- $\mathcal{U}_3$, the uniform distribution over the ternary set $\{\pm 1, 0\}$.
- $\mathcal{U}_q$, the uniform distribution over $\mathbb{Z}_q$.
- $\mathcal{ZO}(\rho)$, a distribution over the ternary set $\{0, \pm 1\}$ with probability $\rho/2$ for ± 1 and probability $1 - \rho$ for 0 with $\rho \in [0, 1]$.

If we choose $a \in R$ from the distributions above, the random variable $a(\zeta)$ has variance $V = \phi(m) \cdot V_a$, where V_a is the variance of each coefficient in a and it is bounded

$$||a||^{can} \leq D\sigma\sqrt{\phi(m)} = D\sqrt{\phi(m) \cdot V_a}, \tag{1}$$

for some D [9]. Moreover, the probability that the variable a exceeds its standard deviation by more than a factor of D is roughly $\mathrm{erfc}(D)$. Thus, we have to choose D large enough to obtain a reasonable failure probability. Specifically, $\mathrm{erfc}(6) \approx 2^{-55}$, $\mathrm{erfc}(5) \approx 2^{-40}$ and $\mathrm{erfc}(4.5) \approx 2^{-32}$.

If $a, b \in R$ are chosen independently randomly and γ is a constant, the following holds for the variances [10]:

- $V_{a+b} = V_a + V_b$.
- $V_{\gamma a} = \gamma^2 V_a$.
- $V_{ab} = \phi(m) V_a V_b$.

Thus, to study the variance of $||a||^{can}$, we have to study the variance V_a of each coefficient a_i of a. Specifically,

$$\begin{array}{llll} a_i \in \mathcal{U}_q & \Rightarrow V_a \approx q^2/12, & a_i \in \mathcal{U}_3 & \Rightarrow V_a = 2/3, \\ a_i \in \mathcal{DG}_q(\sigma^2) & \Rightarrow V_a = \sigma^2, & a_i \in \mathcal{ZO}(\rho) & \Rightarrow V_a = \rho. \end{array} \tag{2}$$

As in [9], we assume that messages behave as if selected uniformly at random from $\mathcal{U}_t$. Thanks to Equations (1) and (2), we have that

$$||m||^{can} \leq Dt\sqrt{\phi(m)/12}. \tag{3}$$

Lattices and Hermite Factor Let $B = (\mathbf{b}_1, \ldots, \mathbf{b}_k)$ be linearly independent vectors in $\mathcal{R}^n$, then the *lattice* $\mathcal{L}(B)$ generated by the *base* B is defined by

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

The dimension k of a lattice $\mathcal{L} \subset \mathcal{R}^n$ is called *rank*. The *volume* (or *determinant*) of $\mathcal{L}$ is defined as $\mathsf{Vol}(\mathcal{L}) = \sqrt{\det(B^t B)}$. In the special case that $\mathcal{L}$ is a full rank lattice, i.e. when $k = n$, we have that $\mathsf{Vol}(\mathcal{L}) = |\det(B)|$. Finally, we can define the *Hermite factor* δ_0^k as

$$\delta_0^k = ||\mathbf{b}_1||/\mathsf{Vol}(\mathcal{L})^{1/k} \tag{4}$$

where $\mathbf{b}_1$ is the shortest vector in the reduced base B of the lattice $\mathcal{L}$. The factor δ_0 is called the *root Hermite factor*.

2.3 Security of RLWE-Based Schemes

The LWE problem consists of finding the secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, given $\mathbf{b} \in \mathbb{Z}_q^m$ and $A \in (\mathbb{Z}_q)^{m \times n}$ such that $A\mathbf{s} + \mathbf{e} = \mathbf{b} \bmod q$, where $\mathbf{e} \in \mathbb{Z}_q^m$ is sampled from the error distribution χ_e. The security of LWE-based schemes depends on the intractability of this problem and attacks on these schemes are based on finding efficient algorithms to solve them [22]. In [5], the authors presented three different methodologies to solve the LWE problem and the central part of two of them is based on lattice reduction. Namely, starting from a bad (i.e. long) lattice basis, find a better (i.e. reduced and more orthogonal) basis.

The most well-known lattice reduction algorithm used in practice is BKZ (block Korkin-Zolotarev reduction) due to Schnorr and Euchner [27]. In these algorithms, the time complexity and the outcome quality (i.e. the orthogonality of the reduced basis) is characterised by the Hermite factor [13]. Specifically, the run time of the BKZ algorithm is higher when the root-Hermite factor δ_0 is smaller [27]. This result is also supported by a realistic estimation provided in [5], where the authors show that the log of the time complexity to get a root-Hermite factor δ_0 with BKZ is

$$\log(t_{BKZ})(\delta_0) = \Omega\left(-\frac{\log(\log \delta_0)}{\log \delta_0}\right) \tag{5}$$

if calling the SVP oracle costs $2^{O(\beta)}$, where β is the the block-size of BKZ algorithm.

2.4 DCRT Representation

The DCRT changes the representation of the polynomials. This also influences the computations itself and thus slight adaptations to the bounds have to be made. In the following, we will briefly explain the DCRT representation and adjust the error bounds accordingly.

To represent polynomials in the DCRT representation, we need to apply two concepts based on the Chinese Remainder Theorem (CRT): the residue number system (RNS) and the Number Theoretic Transform (NTT). The residue number system (RNS) decomposes integers in $\mathbb{Z}_q$ into smaller integers $\mathbb{Z}_{p_i}$ for $q = \prod p_i$. For pairwise coprime p_i, we define a ring isomorphism $\mathbb{Z}_q \cong \mathbb{Z}_{p_1} \times \ldots \times \mathbb{Z}_{p_k}$ with

$$x \bmod q \mapsto (x \bmod p_1, \ldots, x \bmod p_k).$$

In the context of BGV, we decompose a polynomial in $\mathcal{R}_q$ into smaller polynomials in $\mathcal{R}_{p_1} \times \ldots \times \mathcal{R}_{p_k}$.

The Number Theoretic Transform (NTT) and its inverse, the INTT, transform a polynomial to a point-wise representation such that

$$\mathrm{INTT}(\mathrm{NTT}(a) \odot \mathrm{NTT}(b)) = a \cdot b$$

where $\odot$ denotes the point-wise multiplication of the transformed polynomials. This significantly reduces the cost of polynomial multiplication from $\mathcal{O}(n^2)$ to

$\mathcal{O}(n \log n)$, the running time of the NTT. Mathematically, the NTT evaluates the polynomial in each of the m-th roots of unity ζ^j, namely, it decomposes the polynomial into linear terms modulo $(x - \zeta^j)$. For a full definition, we refer the interested reader to [28].

2.5 The BGV Scheme

The BGV scheme is state-of-the-art FHE scheme based on the RLWE hardness assumption. As is common with RLWE-based schemes, implementations of BGV use the DCRT representation for polynomials (see Section 2.4).

Usually, the BGV scheme is used for leveled circuits, that is circuits with a somewhat low multiplicative depth as bootstrapping is very expensive [15]. Hence, we will focus on the leveled version of the BGV scheme in this work.

In the following, we recall the definitions and compute the noise analysis for encryption and the schemes arithmetic operations collecting previous studies [10, 18]. For modulus switching and key switching, we will provide only the definitions and noise bounds, including the RNS variants, and provide a thorough noise analysis later in Section 3. This extends previous work based on the canonical embedding norm with a thourough analysis for an arbitrary plaintext modulus in combination with the RNS.

Key Generation, Encryption & Decryption

KeyGen(λ)

Define parameters and distributions with respect to λ.
Sample $s \leftarrow \chi_s$, $a \leftarrow \mathcal{U}_{q_L}$ and $e \leftarrow \chi_e$.
Output

$$\mathsf{sk} = s \text{ and } \mathsf{pk} = (b, a) \equiv (-a \cdot s + te, a) \pmod{q_L}.$$

Enc$_{\mathsf{pk}}(m)$

Receive plaintext $m \in \mathcal{R}_t$ for $\mathsf{pk} = (b, a)$.
Sample $u \leftarrow \chi_s$ and $e_0, e_1 \leftarrow \chi_e$.
Output $\mathfrak{c} = (\mathbf{c}, L, \nu_{\mathsf{clean}})$ with

$$\mathbf{c} = (c_0, c_1) \equiv (b \cdot u + te_0 + m, a \cdot u + te_1) \pmod{q_L}.$$

Dec$_{\mathsf{sk}}(\mathfrak{c})$

Receive extended ciphertext $\mathfrak{c} = (\mathbf{c}, \ell, \nu)$ for $\mathsf{sk} = s$. Decrypt with

$$c_0 + c_1 \cdot s \equiv m + te \pmod{q_\ell}.$$

Output $m \equiv m + te \bmod t$.

Let $\mathfrak{c} = (\mathbf{c}, \ell, \nu)$ be the *extended ciphertext*, where $\mathbf{c}$ is a ciphertext, ℓ denotes the level and ν the *critical quantity* of $\mathbf{c}$. The critical quantity is defined as the

polynomial $\nu = [c_0 + c_1 \cdot s]_{q_\ell}$ for the associated level ℓ [9]. The critical quantity of a ciphertext $\mathbf{c}$ defines whether $\mathbf{c}$ can be correctly decrypt. Specifically, if the the canonical embedding norm of ν is below a given bound the decryption algorithm works. Otherwise, the plaintext cannot be recovered due to an excessive noise growth.

To understand the error growth and thus analyze the critical quantity ν for each *extended ciphertext* $\mathfrak{c} = (\mathbf{c}, \ell, \nu)$, we apply the decryption algorithm. The following shows the decryption of a ciphertext after an encryption:

$$\begin{aligned} c_0 + c_1 \cdot s &\equiv (-a \cdot s + te) \cdot u + te_0 + m + (a \cdot u + te_1) \cdot s && \pmod{q_L} \\ &\equiv m + t(e \cdot u + e_1 \cdot s + e_0) && \pmod{q_L}. \end{aligned}$$

In general, decryption is correct as long as the error does not wrap around the modulus q_ℓ, that is $||\nu||_\infty \leq c_m ||\nu||^{can} < q_\ell/2$. Note that applying decryption is equivalent to evaluating the ciphertext $\mathbf{c}$ as polynomial in s, that is $c_0 + c_1 \cdot s \equiv \nu \bmod q_\ell$. In the following, we will often use this polynomial representation of a ciphertext to proof correctness of an algorithm or operation.

We derive the bounds for each operation using the canonical embedding norm (Section 2.2). For the encryption operation, we use Equations (2) and (3)

$$\begin{aligned} ||[c_0 + c_1 \cdot s]_{q_\ell}||^{can} &\leq D\sqrt{\phi(m) V_{[c_0 + c_1 \cdot s]_{q_\ell}}} = D\sqrt{\phi(m)\left(V_m + t^2 V_{e \cdot u + e_1 \cdot s + e_0}\right)} \\ &\leq D\sqrt{\phi(m)\left(V_m + t^2(\phi(m) V_e V_u + \phi(m) V_{e_1} V_s + V_{e_0})\right)}. \end{aligned}$$

Namely,

$$\mathsf{B}_{\mathsf{clean}} = Dt\sqrt{\phi(m)\left(1/12 + 2\phi(m) V_e V_s + V_e\right)}. \tag{6}$$

Addition, Multiplication & Constant Multiplication

Add($\mathfrak{c}, \mathfrak{c}'$)
Receive extended ciphertexts $\mathfrak{c} = (\mathbf{c}, \ell, \nu)$ and $\mathfrak{c}' = (\mathbf{c}', \ell, \nu')$.
Output $(\mathbf{c} + \mathbf{c}', \ell, \nu_{\mathsf{add}})$.

Mul($\mathfrak{c}, \mathfrak{c}'$)
Receive extended ciphertexts $\mathfrak{c} = (\mathbf{c}, \ell, \nu)$ and $\mathfrak{c}' = (\mathbf{c}', \ell, \nu')$.
Output $((c_0 \cdot c'_0, c_0 \cdot c'_1 + c_1 \cdot c'_0, c_1 \cdot c'_1), \ell, \nu_{\mathsf{mul}})$.

MulConst($\alpha, \mathfrak{c}$)
Receive constant polynomial $\alpha \in \mathcal{R}_t$ and extended ciphertext $\mathfrak{c} = (\mathbf{c}, \ell, \nu)$.
Output $(\alpha \cdot \mathbf{c}, \ell, \nu_{\mathsf{const}})$.

As long as the bound on each critical quantity stays below the decryption threshold, correctness follows with

$$\begin{aligned}
&\nu_{\mathsf{add}} = \nu + \nu' = [c_0 + c_1 \cdot s]_{q_\ell} + [c'_0 + c'_1 \cdot s]_{q_\ell} \equiv m + m' \bmod t \\
&\quad \Rightarrow ||\nu_{\mathsf{add}}||^{can} \leq ||\nu||^{can} + ||\nu'||^{can} \\
&\nu_{\mathsf{mul}} = \nu \cdot \nu' = [c_0 + c_1 \cdot s]_{q_\ell} \cdot [c'_0 + c'_1 \cdot s]_{q_\ell} \equiv m \cdot m' \bmod t \\
&\quad \Rightarrow ||\nu_{\mathsf{mul}}||^{can} \leq ||\nu||^{can} ||\nu'||^{can} \\
&\nu_{\mathsf{const}} = \alpha \cdot \nu = \alpha \cdot [c_0 + c_1 \cdot s]_{q_\ell} \equiv \alpha \cdot m \bmod t \\
&\quad \Rightarrow ||\nu_{\mathsf{const}}||^{can} \leq ||\alpha||^{can} ||\nu||^{can} \leq \mathsf{B}_{\mathsf{const}} \, ||\nu||^{can} = Dt\sqrt{\phi(m)/12}||\nu||^{can}. \quad (7)
\end{aligned}$$

Here, we also consider the constant α to be uniformly distributed in $\mathcal{R}_t$. Note that the output of the multiplication is still a polynomial in s, but of degree 2. We will later define key switching (see Section 2.5) to modify a ciphertext polynomial $c_0 + c_1 \cdot s + c_2 \cdot s^2$ back to another polynomial $c'_0 + c'_1 \cdot s$ encrypting the same plaintext.

Modulus Switching Modulus switching reduces the associated level and the critical quantity for a ciphertext, enabling leveled homomorphic computations.

The idea is to switch from a ciphertext modulus q_ℓ to a ciphertext modulus $q_{\ell'} = q_{\ell-k}$ for some $k \in \mathbb{Z}$. We thus multiply the ciphertext by $\frac{q_{\ell'}}{q_\ell}$, roughly reducing the error by the same factor. But, as we need to output a valid ciphertext, we add a small correction term $\boldsymbol{\delta}$ that (i) only influences the error, that is being a multiple of t, i.e., $\boldsymbol{\delta} \equiv 0 \bmod t$, and (ii) modifies the ciphertext to be divisible by $q_\ell/q_{\ell'}$, i.e., $\boldsymbol{\delta} \equiv -\mathbf{c} \bmod \frac{q_\ell}{q_{\ell'}}$.

ModSwitch($\mathfrak{c}, \ell'$)

Receive extended ciphertext $\mathfrak{c} = (\mathbf{c}, \ell, \nu)$ and target level $\ell' = \ell - k$.
Set $\boldsymbol{\delta} = t[-\mathbf{c}t^{-1}]_{q_\ell/q_{\ell'}}$ and

$$\mathbf{c}' = \frac{q_{\ell'}}{q_\ell}(\mathbf{c} + \boldsymbol{\delta}) \pmod{q_{\ell'}}.$$

Output $(\mathbf{c}', \ell', \nu_{\mathsf{ms}})$.

For $k = 1$, $\frac{q_{\ell'}}{q_\ell} = \frac{1}{p_\ell}$. First, we want to show the correctness of modulus switching. Let $[c_0 + c_1 \cdot s]_{q_\ell} = c_0 + c_1 \cdot s - kq_\ell$ for some $k \in \mathbb{Z}$. For the same k, let

$$\begin{aligned}
[c'_0 + c'_1 \cdot s]_{q_{\ell'}} &= c'_0 + c'_1 \cdot s - kq_{\ell'} \\
&= \frac{1}{p_\ell}(c_0 + c_1 \cdot s + \delta_0 + \delta_1 \cdot s) - kq_{\ell'} \\
&= \frac{1}{p_\ell}([c_0 + c_1 \cdot s]_{q_\ell} + kq_\ell + \delta_0 + \delta_1 \cdot s) - kq_{\ell'} \\
&= \frac{1}{p_\ell}([c_0 + c_1 \cdot s]_{q_\ell} + \delta_0 + \delta_1 \cdot s) \\
&\equiv p_\ell^{-1} m \bmod t.
\end{aligned}$$

Note that we actually decrypt to the plaintext $p_\ell^{-1} m \bmod t$, but we can multiply a plaintext by p_ℓ either before encryption or after decryption. This issue does not exist for $p_\ell \equiv 1 \bmod t$, but finding such p_ℓ can be difficult in practice.

The error after the modulus switching is bounded by

$$||\nu_{\mathsf{ms}}||^{can} \equiv |||[c_0' + c_1' \cdot s]_{q_{\ell'}}||^{can} \leq \frac{1}{p_\ell} \left(||\nu||^{can} + ||\delta_0 + \delta_1 \cdot s||^{can} \right).$$

As $V_{\delta_i} = V_{tp_\ell} = \frac{t^2 p_\ell^2}{12}$, and thus $V_{\delta_0 + \delta_1 \cdot s} = \frac{t^2 p_\ell^2}{12}(1 + \phi(m) V_s)$, we have

$$||\nu_{\mathsf{ms}}||^{can} \leq \frac{1}{p_\ell}(||\nu||^{can} + D\sqrt{\phi(m) V_{\delta_0 + \delta_1 \cdot s}}) = \frac{1}{p_\ell}||\nu||^{can} + \mathsf{B}_{\mathsf{scale}},$$

with

$$\mathsf{B}_{\mathsf{scale}} = Dt\sqrt{\frac{\phi(m)}{12}(1 + \phi(m) V_s)}. \tag{8}$$

Note that, decryption is correct as long as $||\nu||^{can} < \frac{q_\ell}{2c_m} - p_\ell \mathsf{B}_{\mathsf{scale}}$.

For $k > 1$ in the DCRT representation, we apply a *fast base extension* from $q_\ell / q_{\ell'}$ to $q_{\ell'}$ for $\boldsymbol{\delta}$. For two RNS bases $q = \prod_{i=1}^{\kappa} q_i$ and $q' = \prod_{j=1}^{\kappa'} q_j'$ and a polynomial α, it is defined as

$$\text{BaseExt}(\alpha, q, q') = \sum_{i=1}^{\kappa} \left[\alpha \frac{q_i}{q} \right]_{q_i} \frac{q}{q_i} \pmod{q_j'} \quad \forall j \in \{1, ..., \kappa'\}. \tag{9}$$

This outputs $\alpha + qu$ in the RNS base q'. We analyze the impact of the error in Section 3.

Key Switching Key switching is used for (i) reducing the degree a ciphertext polynomial, usually the output of a multiplication, or (ii) changing the key after a rotation. For a multiplication, we convert the ciphertext term $c_2 \cdot s^2$ to a polynomial $c_0^{\mathsf{ks}} + c_1^{\mathsf{ks}} \cdot s$ and for a rotation, we convert the ciphertext term $c_1 \cdot \text{rot}(s)$ to a polynomial $c_0^{\mathsf{ks}} + c_1^{\mathsf{ks}} \cdot s$. In the following, we will only analyze multiplication and more specifically, we will output $\mathbf{c}' = (c_0 + c_0^{\mathsf{ks}}, c_1 + c_1^{\mathsf{ks}})$ and denote the ciphertext term we want to remove by c_2. This also covers rotations as one only has to consider the term we want to remove as c_1 and an output of $(c_0 + c_0^{\mathsf{ks}}, c_1^{\mathsf{ks}})$. More specifically, we again make use of the RLWE hardness assumption to hide s^2 using s. Decryption with s "unboxes" s^2 and applies it to the ciphertext term c_2. In the following, we provide the general algorithms for key switching:

KeySwitchGen(s, s^2)

Receive secret key s^2 and secret key target s.
Sample $a \leftarrow \mathcal{U}_{q_L}$, $e \leftarrow \chi_e$.
Output key switching key

$$\mathsf{ks} = (\mathsf{ks}_0, \mathsf{ks}_1) \equiv (-a \cdot s + te + s^2, a) \pmod{q_L}.$$

KeySwitch($\mathsf{ks}, \mathfrak{c}$)

Receive extended ciphertext $\mathfrak{c} = (\mathbf{c}, \ell, \nu)$ and key switching key ks.
Switch key for $c_0 + c_1 \cdot s + c_2 \cdot s^2$ with

$$\mathbf{c}' \equiv (c_0 + c_2 \cdot \mathsf{ks}_0, c_1 + c_2 \cdot \mathsf{ks}_1) \pmod{q_\ell}.$$

Output $(\mathbf{c}', \ell, \nu_{\mathsf{ks}})$.

Since q_ℓ divides q_L, $[\mathsf{ks}]_{q_\ell}$ is a valid key switching key with respect to q_ℓ and thus

$$\begin{aligned} c_0' + c_1' \cdot s &\equiv c_0 + c_2 \cdot \mathsf{ks}_0 + (c_1 + c_2 \cdot \mathsf{ks}_1) \cdot s && \pmod{q_\ell} \\ &\equiv c_0 - c_2 \cdot a \cdot s + c_2 \cdot te + c_2 \cdot s^2 + c_1 \cdot s + c_2 \cdot a \cdot s && \pmod{q_\ell} \\ &\equiv c_0 + c_1 \cdot s + c_2 \cdot s^2 + tc_2 \cdot e && \pmod{q_\ell}. \end{aligned}$$

Thus, the error after the key switching algorithm is bounded by

$$||\nu_{\mathsf{ks}}||^{can} = ||[c_0' + c_1' \cdot s]_{q_\ell}||^{can} \leq ||\nu||^{can} + tc_2 \cdot e.$$

Unfortunately, the error after the key switching algorithm grows too much with the term $tc_2 \cdot e$ and thus, several variants exist to reduce its growth. This work considers the three main variants: the Brakerski Vaikuntanathan (BV) variant, the Gentry Halevi Smart (GHS) variant, and the Hybrid variant.

BV Variant The BV variant decomposes c_2 with respect to a base β to reduce the error growth [7]. For polynomials a and b and $l = \lfloor \log_\beta q_\ell \rceil + 1$, we define

$$\begin{aligned} \mathcal{D}_\beta(a) &= ([a]_\beta, [\lfloor a/\beta \rfloor]_\beta, \ldots, [\lfloor a/\beta^{l-1} \rfloor]_\beta) \\ \mathcal{P}_\beta(b) &= ([b]_{q_\ell}, [b\beta]_{q_\ell}, \ldots, [b\beta^{l-1}]_{q_\ell}). \end{aligned}$$

It follows that, for any $a, b \in \mathcal{R}_{q_\ell}$, we have $\langle \mathcal{D}_\beta(a), \mathcal{P}_\beta(b) \rangle \equiv a \cdot b \bmod q_\ell$[19].

KeySwitchGen$^{\mathsf{BV}}(s, s^2)$

Receive secret key s' and secret key target s.
Sample $\mathbf{a} \leftarrow \mathcal{U}_{q_L}^l$, $\mathbf{e} \leftarrow \chi_e^l$.
Output key switching key

$$\mathsf{ks}^{\mathsf{BV}} = (\mathbf{ks}_0^{\mathsf{BV}}, \mathbf{ks}_1^{\mathsf{BV}}) = (-\mathbf{a} \cdot s + t\mathbf{e} + \mathcal{P}_\beta(s^2), \mathbf{a}) \pmod{q_L}.$$

KeySwitch$^{\mathsf{BV}}(\mathbf{ks}^{\mathsf{BV}}, \mathfrak{c})$

Receive extended ciphertext $\mathfrak{c} = (\mathbf{c}, \ell, \nu)$ and key switching key $\mathsf{ks}^{\mathsf{BV}}$.
Switch key for $c_0 + c_1 \cdot s + c_2 \cdot s^2$ with

$$\mathbf{c}' = (c_0 + \langle \mathcal{D}_\beta(c_2), \mathbf{ks}_0^{\mathsf{BV}} \rangle, c_1 + \langle \mathcal{D}_\beta(c_2), \mathbf{ks}_1^{\mathsf{BV}} \rangle) \pmod{q_\ell}.$$

Output $(\mathbf{c}', \ell, \nu_{\mathsf{ks}}^{\mathsf{BV}})$.

The error after the BV key switching is $c'_0+c'_1\cdot s \equiv c_0+c_2\cdot \mathsf{ks}_0^{\mathsf{BV}}+(c_1+c_2\cdot \mathsf{ks}_1^{\mathsf{BV}})\cdot s \pmod{q_\ell}$, namely,

$$||[c_0 + c_1 \cdot s + \langle \mathcal{D}_\beta(c_2), \mathcal{P}_\beta(s^2)\rangle + t\langle \mathcal{D}_\beta(c_2), \mathbf{e}\rangle]_{q_\ell}||^{can},$$

that is,

$$||\nu_{\mathsf{ks}}^{\mathsf{BV}}||^{can} = ||[c'_0 + c'_1 \cdot s]_{q_\ell}||^{can} \leq ||\nu||^{can} + ||t\langle \mathcal{D}_\beta(c_2), \mathbf{e}\rangle||^{can}.$$

Since $t\langle \mathcal{D}_\beta(c_2), \mathbf{e}\rangle = t\sum_{i=0}^{l-1}[\lfloor c_2/\beta^i \rfloor]_\beta \cdot e_i = t\sum_{i=0}^{l-1} \tilde{\beta}_i \cdot e_i$, we have

$$V_{t\cdot\langle \mathcal{D}_\beta(c_2),\mathbf{e}\rangle} = t^2 l\phi(m) V_{\tilde{\beta}_i} V_{e_i}.$$

We can assume that $\tilde{\beta}_i$ behaves like a uniform polynomial drawn from $\mathcal{U}_\beta$. So

$$||t \cdot \langle \mathcal{D}_\beta(c_2), \mathbf{e}\rangle||^{can} \leq D\sqrt{\phi(m)t^2 l\phi(m)\frac{\beta^2}{12}V_{e_i}} = Dt\phi(m)\beta\sqrt{l\frac{V_e}{12}}.$$

Finally, we have $l = \sqrt{\lfloor \log_\beta(q_\ell)\rfloor + 1} \sim \sqrt{\log_\beta(q_\ell)}$ and can set

$$||\nu_{\mathsf{ks}}^{\mathsf{BV}}||^{can} \leq ||\nu||^{can} + \beta\sqrt{\log_\beta(q_\ell)}\mathsf{B}_{\mathsf{ks}},$$

where

$$\mathsf{B}_{\mathsf{ks}} = Dt\phi(m)\sqrt{V_e/12}. \tag{10}$$

BV-RNS Variant For the BV-RNS variant, we additionally define $\mathcal{D}_{p_i}$ and $\mathcal{P}_{p_i}$ not with respect to some digit decomposition β, but to the already existing RNS decomposition which we hence can apply for free during the key switching process.

$$\mathcal{D}_{p_i}(a) = \left(\left[a\left(\frac{q_\ell}{p_0}\right)^{-1}\right]_{p_0}, \ldots, \left[a\left(\frac{q_\ell}{p_\ell}\right)^{-1}\right]_{p_\ell}\right)$$
$$\mathcal{P}_{p_i}(b) = \left(\left[b\frac{q_\ell}{p_0}\right]_{q_\ell}, \ldots, \left[b\frac{q_\ell}{p_\ell}\right]_{q_\ell}\right).$$

Since this itself does not reduce the error enough, we apply both decompositions at the same time with $\langle \mathcal{D}_\beta(\mathcal{D}_{p_i}(a)), \mathcal{P}_{p_i}(\mathcal{P}_\beta(b))\rangle = a \cdot b \bmod q_\ell$. We analyze the noise growth in Section 3.

GHS Variant The GHS variant [15] switches to a bigger ciphertext modulus $Q_\ell = q_\ell P$ with P and q coprime. Then, key switching takes places in $\mathcal{R}_{Q_\ell}$ and, by modulus switching back down to q_ℓ, the error is reduced again. As a tradeoff, we have to make sure that our RLWE instances are secure with respect to Q_ℓ.

KeySwitchGen$^{\mathsf{GHS}}(s, s^2)$

Receive secret key s^2 and secret key target s.
Sample $a \leftarrow \mathcal{U}_{Q_L}$, $e \leftarrow \chi_e$.
Output key switching key

$$\mathsf{ks}^{\mathsf{GHS}} = (\mathsf{ks}_0^{\mathsf{GHS}}, \mathsf{ks}_1^{\mathsf{GHS}}) \equiv (-a \cdot s + te + Ps^2, a) \pmod{Q_L}.$$

KeySwitch$^{\mathsf{GHS}}(\mathbf{ks}, \mathfrak{c})$

Receive extended ciphertext $\mathfrak{c} = (\mathbf{c}, \ell, \nu)$ and key switching key $\mathsf{ks}^{\mathsf{GHS}}$.
For $c_0 + c_1 \cdot s + c_2 \cdot s^2$, switch key with

$$\mathbf{c}' \equiv (Pc_0 + c_2 \cdot \mathsf{ks}_0^{\mathsf{GHS}}, Pc_1 + c_2 \cdot \mathsf{ks}_1^{\mathsf{GHS}}) \bmod Q_\ell.$$

Set $\boldsymbol{\delta} = t[-\mathbf{c}'t^{-1}]_P$, modulus switch back with

$$\mathbf{c}'' = \frac{1}{P}(\mathbf{c}' + \boldsymbol{\delta}) \pmod{q_\ell}.$$

Output $(\mathbf{c}'', \ell, \nu_{\mathsf{ks}}^{\mathsf{GHS}})$.

Since we use modulus switching, showing correctness is similar in most aspects. For some $k \in \mathbb{Z}$, let $[c_0' + c_1' \cdot s]_{Q_\ell} = P[c_0 + c_1 \cdot s + c_2 \cdot s^2]_{q_\ell} + tc_2 \cdot e - kQ_\ell$.

$$\begin{aligned}[c_0'' + c_1'' \cdot s]_{q_\ell} &= [c_0 + c_1 \cdot s + c_2 \cdot s^2]_{q_\ell} + \frac{tc_2 \cdot e + \delta_0 + \delta_1 \cdot s}{P} \\ &\equiv m \bmod t.\end{aligned}$$

We suppose that c_2 behaves like a uniform polynomial samples from $\mathcal{U}_{q_\ell}$ and, as before, $[-\mathbf{c}t^{-1}]_P$ behaves like a uniform polynomial samples from $\mathcal{U}_P$. Then,

$$\begin{aligned}||\nu_{\mathsf{ks}}^{\mathsf{GHS}}||^{can} &\leq |||[c_0 + c_1 \cdot s + c_2 \cdot s^2]_{q_\ell}||^{can} + \frac{||tc_2 \cdot e + \delta_0 + \delta_1 \cdot s||^{can}}{P} \\ &= ||\nu||^{can} + \frac{D\sqrt{\phi(m)(t^2 q_\ell^2 \frac{V_e}{12} + t^2 P^2 \frac{1}{12}(1 + \phi(m)V_s))}}{P} \\ &\leq ||\nu||^{can} + \frac{q_\ell}{P}\mathsf{B}_{\mathsf{ks}} + \mathsf{B}_{\mathsf{scale}},\end{aligned}$$

where $\mathsf{B}_{\mathsf{scale}}$ and B_{ks} are as in Equations (8) and (10), respectively. Decryption, and thus key switching, is correct as long as $||\nu||^{can} < \frac{q_\ell}{2c_m} - \frac{q_\ell}{P}\mathsf{B}_{\mathsf{ks}} - \mathsf{B}_{\mathsf{scale}}$.

GHS-RNS Variant For the RNS variant of GHS, we set $P = \prod_{j=1}^{k} P_j$ such that $P_i \equiv 1 \mod m$ and extend $\boldsymbol{\delta}$ from P to q_ℓ (see Equation (9)). Additionally, we extend c_2 from q_ℓ to Q_ℓ. We analyze the noise growth in Section 3.

Hybrid Variant The Hybrid variant combines the BV and GHS variants [15]. In the following, we use the same notation from the variants as before.

KeySwitchGen$^{\mathsf{Hybrid}}(s, s^2)$

Receive secret key s^2 and secret key target s.
Sample $\mathbf{a} \leftarrow \mathcal{U}^l_{Q_L}$, $\mathbf{e} \leftarrow \chi^l_e$.
Output key switching key

$$\mathsf{ks}^{\mathsf{Hybrid}} = (\mathsf{ks}_0^{\mathsf{Hybrid}}, \mathsf{ks}_1^{\mathsf{Hybrid}}) \equiv (-\mathbf{a} \cdot s + t\mathbf{e} + P\,\mathcal{P}_\beta(s^2), \mathbf{a}) \pmod{Q_L}.$$

KeySwitch$^{\mathsf{Hybrid}}(\mathbf{ks}^{\mathsf{Hybrid}}, \mathfrak{c})$

Receive extended ciphertext $\mathfrak{c} = (\mathbf{c}, \ell, \nu)$ and key switching key $\mathsf{ks}^{\mathsf{Hybrid}}$.
For $c_0 + c_1 \cdot s + c_2 \cdot s^2$, switch key with

$$\mathbf{c}' \equiv (Pc_0 + \langle \mathcal{D}_\beta(c_2), \mathsf{ks}_0^{\mathsf{Hybrid}} \rangle, Pc_1 + \langle \mathcal{D}_\beta(c_2), \mathsf{ks}_1^{\mathsf{Hybrid}} \rangle) \bmod Q_\ell.$$

Set $\boldsymbol{\delta} = t[-\mathbf{c}'t^{-1}]_P$, modulus switch back with

$$\mathbf{c}'' = \frac{1}{P}(\mathbf{c}' + \boldsymbol{\delta}) \pmod{q_\ell}.$$

Output $(\mathbf{c}'', \ell, \nu_{\mathsf{ks}}^{\mathsf{Hybrid}})$.

Correctness follows by combining the proofs of each variant. The bounds also follow similarly, since before to scale down we have $\nu' = \nu P + \beta l \mathsf{B}_{\mathsf{ks}}$, where B_{ks} is as an Equation (10). Thus, the error after the modulus switching procedure is bounded by $\frac{q_\ell}{Q_\ell}||\nu'||^{can} + \mathsf{B}_{\mathsf{scale}}$, that is,

$$||\nu_{\mathsf{ks}}^{\mathsf{Hybrid}}||^{can} \leq ||\nu||^{can} + \frac{\beta\sqrt{\log_\beta(q_\ell)}}{P}\mathsf{B}_{\mathsf{ks}} + \mathsf{B}_{\mathsf{scale}},$$

where $\mathsf{B}_{\mathsf{scale}}$ is defined as Equation (8).

Hybrid-RNS Variant. The Hybrid-RNS variant combines the RNS adaptations of each variant. However, instead of decomposing with respect to each single RNS prime, we group the primes into ω chunks of (at most) size $l = \lceil \frac{L}{\omega} \rceil$ and do not apply the decomposition to the base β. Hence, we set $\tilde{q}_i = \prod_{j=il}^{il+l-1} p_j$ and define $\mathcal{D}_{\tilde{q}_i}$ and $\mathcal{P}_{\tilde{q}_i}$ as

$$\mathcal{D}_{\tilde{q}_i}(\alpha) = \left(\left[\alpha \left(\frac{q_\ell}{\tilde{q}_0} \right)^{-1} \right]_{\tilde{q}_0}, \ldots, \left[\alpha \left(\frac{q_\ell}{\tilde{q}_{l-1}} \right)^{-1} \right]_{\tilde{q}_{l-1}} \right)$$

$$\mathcal{P}_{\tilde{q}_i}(\beta) = \left(\left[\beta \frac{q_\ell}{\tilde{q}_0} \right]_{q_\ell}, \ldots, \left[\beta \frac{q_\ell}{\tilde{q}_{l-1}} \right]_{q_\ell} \right).$$

We now have to extend c_2 from each $\tilde{q}_i$ to Q_ℓ instead. We analyze the noise growth in Section 3.

3 Improving the Parameter Generation Process

In this section, we provide our improvements to the parameter generation process. First, we offer new bounds for modulus switching and key switching in the DCRT representation. These bounds enable correct analysis for essential BGV operations, such as rotations supporting different methods for key switching. Afterward, we suggest an improvement to circuit models in general and define new circuit models. We analyze these models with our newly improved bounds and provide closed formulas to compute the individual primes. Finally, we introduce the empirically derived, closed security formula for our parameter generator and shortly describe the generator itself.

3.1 Modulus Switching

For modulus switching, we can either scale by a single modulus or by multiple moduli. When scaling by a single modulus, the bound $\mathsf{B}_{\mathsf{scale}}$ is as in Equation (8).

When scaling by $\jmath > 1$ moduli, however, we have to adjust our bound due to the base extension of $t[-\mathbf{c}t^{-1}]_{q_\ell/q_{\ell'}}$ from $q_\ell/q_{\ell'}$ to $q_{\ell'}$ with $\ell' = \ell - \jmath$.

Let $[c_0 + c_1 \cdot s]_{q_\ell} = c_0 + c_1 \cdot s - \kappa q_\ell$ for some $\kappa \in \mathbb{Z}$. For the same κ, let $[c'_0 + c'_1 \cdot s]_{q_{\ell'}} = c'_0 + c'_1 \cdot s - \kappa q_{\ell'}$, then

$$\begin{aligned}[c'_0 + c'_1 \cdot s]_{q_{\ell'}} &= c'_0 + c'_1 \cdot s - \kappa q_{\ell'} \\ &= \frac{q_{\ell'}}{q_\ell}(c_0 + c_1 \cdot s + \delta_0 + \delta_1 \cdot s) - \kappa q_{\ell'} \\ &= \frac{q_{\ell'}}{q_\ell}([c_0 + c_1 \cdot s]_{q_\ell} + \delta_0 + \delta_1 \cdot s).\end{aligned}$$

Considering Equation (9), we extend each δ_i as

$$\delta_i = t\,\mathrm{BaseExt}(-c_i t^{-1}, \frac{q_\ell}{q_{\ell'}}, q_{\ell'}) = t \sum_{i=\ell'+1}^{\ell} \left[-c_i t^{-1} \frac{p_i}{q_\ell/q_{\ell'}}\right]_{p_i} \frac{q_\ell/q_{\ell'}}{p_i} \pmod{q_{\ell'}}$$

Then, the variance V_{δ_i} follows as

$$V_{\delta_i} = t^2 \jmath V_{p_i} \frac{(q_\ell/q_{\ell'})^2}{p_i^2} = t^2 \jmath \frac{(q_\ell/q_{\ell'})^2}{12}.$$

Thus, we introduce a factor of $\jmath$ compared to the non-extended variance of δ_i and the correct bound follows as $||\nu_{\mathsf{ms}}||^{can} \leq \frac{q_{\ell'}}{q_\ell}||\nu||^{can} + \sqrt{\jmath}\mathsf{B}_{\mathsf{scale}}$.

3.2 New DCRT Bounds for Key Switching

For the BV-RNS variant, $\mathcal{D}_{p_i}$ decomposes to each individual modulus and the key switching is bound by

$$\begin{aligned}||\nu_{\mathsf{ks}}^{\mathsf{BV-RNS}}||^{can} &\leq ||\nu||^{can} + ||t\langle \mathcal{D}_{p_i}(c_2), \mathbf{e}\rangle||^{can} \\ &\leq ||\nu||^{can} + \sqrt{\ell+1}\max(p_i)\mathsf{B}_{\mathsf{ks}}.\end{aligned}$$

Assuming $p_i < p_j$ for $i < j$, this simplifies to

$$||\nu_{\mathsf{ks}}^{\mathsf{BV-RNS}}||^{can} \leq ||\nu||^{can} + \sqrt{\ell+1}p_\ell \mathsf{B}_{\mathsf{ks}}.$$

Applying the additional decomposition $\mathcal{D}_\beta$ results in our final bound

$$||\nu_{\mathsf{ks}}^{\mathsf{BV-RNS}}||^{can} \leq ||\nu||^{can} + \sqrt{l+1}\beta\sqrt{\log_\beta p_\ell}\mathsf{B}_{\mathsf{ks}}.$$

For the GHS-RNS variant, we have two additional errors from the base extension: once for extending c_2 from q_ℓ to Q_ℓ and once for extending $\boldsymbol{\delta}$ from P to Q_ℓ. When extending c_2 and multiplying with the key switching key, this results in

$$\begin{aligned} c_0' + c_1's &\equiv Pc_0 + (c_2 + q_\ell u)\cdot \mathsf{ks}_0 + (Pc_1 + (c_2 + q_\ell u)\cdot \mathsf{ks}_1)s \pmod{Q_\ell} \\ &\equiv P[c_0 + c_1 s + c_2 s^2]_{q_\ell} + t(c_2 + q_\ell u)e \pmod{Q_\ell} \end{aligned}$$

increasing the noise to $||\nu'||^{can} \leq P||\nu||^{can} + \sqrt{\ell+1}q_\ell \mathsf{B}_{\mathsf{ks}}$. When extending $\boldsymbol{\delta}$, the additional noise behaves as equivalent to our modulus switching analysis. Thus, for $P = \prod_{j=1}^{k} P_j$, we get a factor of $\sqrt{k}$ and overall

$$||\nu_{\mathsf{ks}}^{\mathsf{GHS-RNS}}||^{can} \leq ||\nu||^{can} + \sqrt{\ell+1}\frac{q_\ell}{P}\mathsf{B}_{\mathsf{ks}} + \sqrt{k}\mathsf{B}_{\mathsf{scale}}.$$

For the Hybrid-RNS variant, we can combine our previous analyses. However, we again have to adjust because we decompose the ciphertext modulus into ω products $\tilde{q}$ and not necessarily to each individual RNS prime. The fast base extension takes place before the dot product, for an upper bound we now consider $\max \tilde{q}_i$ instead of $\max q_i$ leading to

$$||\nu_{\mathsf{ks}}^{\mathsf{Hybrid-RNS}}||^{can} \leq ||\nu||^{can} + \sqrt{\omega(\ell+1)}\frac{\max(\tilde{q}_i)}{P}\mathsf{B}_{\mathsf{ks}} + \sqrt{k}\mathsf{B}_{\mathsf{scale}}.$$

If we again assume $p_i < p_j$ for $i < j$, this simplifies to

$$||\nu_{\mathsf{ks}}^{\mathsf{Hybrid-RNS}}||^{can} \leq ||\nu||^{can} + \sqrt{\omega(\ell+1)}\frac{p_\ell^{\lceil \ell/\omega \rceil}}{P}\mathsf{B}_{\mathsf{ks}} + \sqrt{k}\mathsf{B}_{\mathsf{scale}}.$$

3.3 Modeling the Homomorphic Circuit

We generally split the homomorphic circuit into levels and reduce the ciphertext noise to a base level B using modulus switching. The multiplicative depth M determines the modulus count $L = M + 1$, which can be split into three types: top, middle, and bottom modulus.

- The top modulus p_{L-1} is the first modulus in the prime chain. Before any operation, we reduce the fresh encryption noise $\mathsf{B}_{\mathsf{clean}}$ down to the base noise B using modulus switching. We continue in level $L-2$ with the middle modulus.

- The middle modulus p_ℓ at level $1 \leq \ell \leq L-2$ reduces the noise back to B after the arithmetic operations as defined by the model (see below) using the modulus switching procedure. This reduces the modulus from q_ℓ to $q_{\ell-1}$ until the last modulus $q_0 = p_0$.
- For the bottom modulus p_0, we can still perform all arithmetic operations within our model. However, we do not scale down to B. Instead, this modulus is large enough to perform decryption correctly. Instead of performing a key switching including modulus switching for the final multiplication, we decrypt right after this multiplication using sk and sk^2, reducing the overall amount of operations that are performed.

This work studies the circuit models depicted in Figure 1. After the initial modulus switching, we operate on η ciphertexts c_i in parallel. In our Model 1, we perform one constant multiplication followed by τ rotations. Finally, the ciphertexts are accumulated and used as input to one multiplication before modulus switching is applied. In Model 2, we switch the rotations and constant multiplications, corresponding to the worst possible noise growth. For $\tau = 0$, we refer to the model as Base Model and provide separate formulas as it simplifies analysis and thus also reduces the ciphertext modulus size. For comparison with previous work, we also define the model as used in the OpenFHE library [6, 19]. For the OpenFHE model, the multiplication is before the circuit with an input noise of B^2 for each $\mathbf{c}_i$. Note that within our models, all parameters can be chosen as required by the use case.

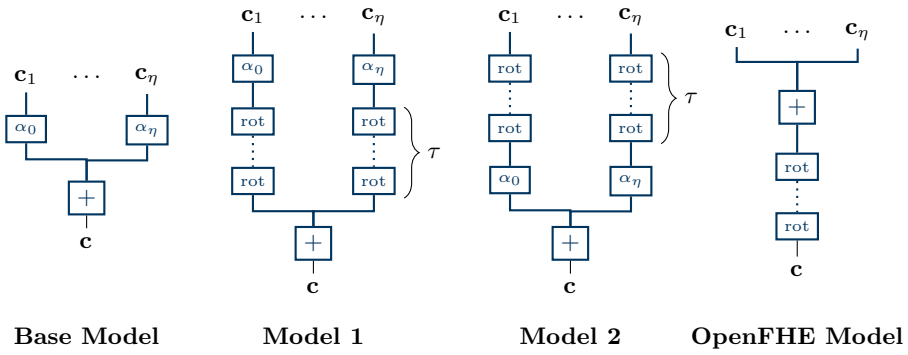

Fig. 1. Our analysis models are depicted as circuits.

3.4 Determining Modulus Sizes

In the following, we determine the modulus size by computing bounds for the individual primes in the modulus chain. Based on our models, we assume that

after each level, we apply modulus switching to go down to a certain noise level B. Given the top modulus, we scale down to B right after the encryption. With the middle modulus, we scale down the level noise back to B after each level. Finally, with the bottom modulus, we perform our leveled computation followed by the decryption.

Middle Modulus. We describe our approach by first analyzing the middle modulus p_ℓ in the base model: considering η ciphertexts $\mathbf{c}_i$ with a starting noise of B, we apply a constant multiplication, thus the noise increments to $\mathsf{B}_{\mathsf{const}}B$, and add up all the ciphertexts, increasing the noise to $\eta\mathsf{B}_{\mathsf{const}}B$. When multiplying two ciphertexts within this model, the noise after the multiplication grows to $(\eta\mathsf{B}_{\mathsf{const}}B)^2$.

We conclude the level by applying a key switching for the multiplication, increasing the noise by ν_{ks}, as well as a modulus switching which scales the noise back down. This results in the inequality

$$\frac{(\eta\mathsf{B}_{\mathsf{const}}B)^2 + \nu_{\mathsf{ks}}}{p_\ell} + \mathsf{B}_{\mathsf{scale}} < B \tag{11}$$

where the key switching noise ν_{ks} depends on the key switching method with

$$\nu_{\mathsf{ks}} = f_0\mathsf{B}_{\mathsf{ks}} + \sqrt{f_1}\mathsf{B}_{\mathsf{scale}} \tag{12}$$

and

$$f_0 = \begin{cases} \beta\sqrt{\log_\beta q_\ell} & \text{for BV} \\ \sqrt{\ell+1}\beta\sqrt{\log_\beta p_\ell} & \text{for BV-RNS} \\ q_\ell/P & \text{for GHS} \\ \sqrt{\ell+1}q_\ell/P & \text{for GHS-RNS} \\ \beta\sqrt{\log_\beta q_\ell}/P & \text{for Hybrid} \\ \sqrt{\omega(\ell+1)}p_\ell^{\lceil L/\omega\rceil}/P & \text{for Hybrid-RNS} \end{cases} \text{ and } f_1 = \begin{cases} 0 & \text{for BV} \\ 0 & \text{for BV-RNS} \\ 1 & \text{for GHS} \\ k & \text{for GHS-RNS} \\ 1 & \text{for Hybrid} \\ k & \text{for Hybrid-RNS} \end{cases}.$$

We similarly model the noise growth in the other models, resulting in the inequalities

$$\begin{aligned} \frac{\eta^2(\mathsf{B}_{\mathsf{const}}B + \tau\nu_{\mathsf{ks}})^2 + \nu_{\mathsf{ks}}}{p_\ell} + \mathsf{B}_{\mathsf{scale}} < B \qquad & \text{for our model 1,} \\ \frac{\eta^2\mathsf{B}_{\mathsf{const}}^2(B + \tau\nu_{\mathsf{ks}})^2 + \nu_{\mathsf{ks}}}{p_\ell} + \mathsf{B}_{\mathsf{scale}} < B \qquad & \text{for our model 2,} \\ \frac{\eta B^2 + (\tau+1)\nu_{\mathsf{ks}}}{p_\ell} + \mathsf{B}_{\mathsf{scale}} < B \qquad & \text{for the OpenFHE model.} \end{aligned} \tag{13}$$

Solving these inequalities then determines a value for B and the sizes of our middle moduli p_ℓ. In the following, we use $\varepsilon = \eta\mathsf{B}_{\mathsf{const}}$ and $\xi = \varepsilon^2$ to simplify notation. If constant multiplications are not required, we set $\mathsf{B}_{\mathsf{const}} = 1$.

Top Modulus. For the top modulus, we simply want to reduce the encryption noise to B using the modulus switching, hence we set

$$\frac{\mathsf{B}_{\mathsf{clean}}}{p_{L-1}} + \mathsf{B}_{\mathsf{scale}} < B \iff p_{L-1} > \frac{\mathsf{B}_{\mathsf{clean}}}{B - \mathsf{B}_{\mathsf{scale}}}. \tag{14}$$

Bottom Modulus. For the bottom modulus $p_0 = q_0$, we do not apply the key switching for the multiplication or the modulus switching afterward while still allowing the computations as defined in the respective circuit models. To ensure correct decryption, we require that $||\nu||_\infty \leq c_m||\nu||^{can} < q_0/2$. For our models, this results in the following bounds for the bottom modulus:

$$\begin{aligned}
&p_0 > 2c_m(\eta\mathsf{B}_{\mathsf{const}}B)^2 && \text{for our base model,}\\
&p_0 > 2c_m\eta^2(\mathsf{B}_{\mathsf{const}}B + \tau\nu_{\mathsf{ks}})^2 && \text{for our model 1,}\\
&p_0 > 2c_m\eta^2\mathsf{B}^2_{\mathsf{const}}(B + \tau\nu_{\mathsf{ks}})^2 && \text{for our model 2,}\\
&p_0 > 2c_m\eta B^2 + \tau\nu_{\mathsf{ks}} && \text{for the OpenFHE model.}
\end{aligned} \tag{15}$$

Thus, we simply choose each bottom modulus larger than the corresponding bound in the respective model.

3.5 Computing the Noise Bound B

In the following, considering the different key-switching procedures, we determine the bound B required to select the size of our individual moduli in the modulus chain. We apply our method exemplary to Model 2 with the worst-case noise growth. The same techniques can, however, be applied to simpler models as well.

Note that for GHS and Hybrid key switching as well as its RNS variants, we can merge the key switching with the modulus switching and directly switch down to a smaller modulus, that is, from Q_ℓ to $q_{\ell-1}$ decreasing the noise by $q_{\ell-1}/Q_\ell = 1/(Pp_\ell)$. Including this optimization, in the specific case of GHS and Hybrid key switching, we have to adjust Equation (13) for Model 2 to

$$\frac{\eta^2\mathsf{B}^2_{\mathsf{const}}(B + \tau\nu_{\mathsf{ks}})^2 + f_0\mathsf{B}_{\mathsf{scale}}}{p_\ell} + \sqrt{f_1'}\mathsf{B}_{\mathsf{scale}} < B \tag{16}$$

where

$$f_1' = \begin{cases} 1 & \text{for GHS, and Hybrid} \\ k+1 & \text{for GHS-RNS, and Hybrid-RNS.} \end{cases}$$

BV key-switching. To compute the noise bound B we have to consider the Equation (13). Note that, since the p_ℓ's have roughly the same size, $f_0^{\mathsf{BV}} \sim f_0^{\mathsf{BV-RNS}}$. Indeed, $f_0^{\mathsf{BV}} = \beta\sqrt{\log_\beta q_\ell} \sim \beta\sqrt{(\ell+1)\log_\beta(p_\ell)}$. Since $\log_\beta(p_\ell) \leq \log_2(p_\ell)$, we can set

$$f_0 = f_0^{\mathsf{BV}} = f_0^{\mathsf{BV-RNS}} = \beta\sqrt{(\ell+1)\log_2(p_\ell)}.$$

Moreover, for p_ℓ big enough, we have that $f_0\mathsf{B}_{\mathsf{ks}}/p_\ell \sim 0$. Then, Equation (13) becomes

$$\frac{\xi B^2}{p_\ell} + \Big(\frac{2\xi\tau f_0\mathsf{B}_{\mathsf{ks}}}{p_\ell} - 1\Big)B + \frac{(\varepsilon\tau f_0\mathsf{B}_{\mathsf{ks}})^2}{p_\ell} + \mathsf{B}_{\mathsf{scale}} < 0.$$

Because of the previous inequality, in the indeterminate B, must have a positive discriminant, we have

$$1 - \frac{4\xi\tau f_0\mathsf{B}_{\mathsf{ks}}}{p_\ell} - \frac{4\xi\mathsf{B}_{\mathsf{scale}}}{p_\ell} \geq 0 \iff p_\ell > 4\xi(\tau f_0\mathsf{B}_{\mathsf{ks}} + \mathsf{B}_{\mathsf{scale}}). \tag{17}$$

Thus we can set

$$p_1 \sim \ldots \sim p_{L-2} \sim 4\xi(3\tau\beta\sqrt{(L-1)\log_2(t\phi(m))}\mathsf{B}_{\mathsf{ks}} + \mathsf{B}_{\mathsf{scale}}), \tag{18}$$

Indeed, considering p_ℓ as in Equation (18), the Equation (17) holds. Moreover, since the discriminant ~ 0, we have

$$B \sim \frac{1}{2\xi/p_\ell} \sim 2(3\tau\beta\sqrt{(L-1)\log_2(t\phi(m))}\mathsf{B}_{\mathsf{ks}} + \mathsf{B}_{\mathsf{scale}}). \tag{19}$$

where $\beta \geq 2$.

We can now compute the bottom modulus p_0 starting from Equation (15). This equation becomes $z > \varepsilon B + \varepsilon\tau\omega\mathsf{B}_{\mathsf{ks}}\sqrt{\log_\omega(2c_m z^2)}$ where $z = \sqrt{q_0/2c_m}$, i.e., $q_0 = 2c_m z^2$. Since $\sqrt{\log_\omega(2c_m)} + \sqrt{2\log_\omega z} > \sqrt{\log_\omega(2c_m z^2)}$, it is enough to prove that

$$z > \varepsilon B + \varepsilon\tau\omega\mathsf{B}_{\mathsf{ks}}\big(\sqrt{\log_\omega(2c_m)} + \sqrt{2\log_\omega z}\big) \tag{20}$$

We claim that this inequality holds for $z = 2\varepsilon B$. Indeed, Equation (20) becomes

$$\frac{\varepsilon B}{\varepsilon\tau\omega\mathsf{B}_{\mathsf{ks}}} - \sqrt{\log_\omega(2c_m)} > \sqrt{2\log_\omega(2\varepsilon B)}.$$

Since B is as in Equation (19), then $2\varepsilon B < \mathsf{B}_{\mathsf{ks}}^3$, so the previous inequality holds since

$$2\sqrt{L-1}\log_\omega(\mathsf{B}_{\mathsf{ks}}) + \frac{2\mathsf{B}_{\mathsf{scale}}}{\tau\omega\mathsf{B}_{\mathsf{ks}}} > \sqrt{6\log_\omega(\mathsf{B}_{\mathsf{ks}})} > \sqrt{2\log_\omega(2\varepsilon B)}.$$

Namely, $p_0 > 8c_m\xi B^2$.

Base Model. When $\tau = 0$, Equation (13) becomes

$$\xi B^2/p_\ell + \beta\sqrt{(\ell+1)\log_2(p_\ell)}\mathsf{B}_{\mathsf{ks}}/p_\ell + \mathsf{B}_{\mathsf{scale}} < B.$$

Following the same argument as before, we must have a positive discriminant and thus we have $p_1 \sim \ldots \sim p_{L-2} \sim 4\xi\mathsf{B}_{\mathsf{scale}}$, $B \sim \frac{p_\ell}{2\xi} \sim 2\mathsf{B}_{\mathsf{scale}}$ and $p_0 > 2c_m\xi B^2 = 8c_m\xi\mathsf{B}_{\mathsf{scale}}^2$.

GHS and Hybrid key-switching. Equation (16) becomes

$$\frac{\xi B^2}{p_\ell} + \frac{2\xi\tau}{p_\ell}\Big(f_0\mathsf{B}_{\mathsf{ks}} + \sqrt{f_1}\mathsf{B}_{\mathsf{scale}}\Big)B + \frac{\xi\tau^2}{p_\ell}\Big(f_0\mathsf{B}_{\mathsf{ks}} + \sqrt{f_1}\mathsf{B}_{\mathsf{scale}}\Big)^2 + \frac{f_0}{p_\ell}\mathsf{B}_{\mathsf{ks}} + \sqrt{f_1'}\mathsf{B}_{\mathsf{scale}} < B. \tag{21}$$

To solve this inequality in B, we follow the idea of Gentry *et al.* [15]. Let

$$R_\ell = \frac{\xi\tau^2}{p_\ell}\Big(f_0\mathsf{B}_{\mathsf{ks}} + \sqrt{f_1}\mathsf{B}_{\mathsf{scale}}\Big)^2 + \frac{f_0}{p_\ell}\mathsf{B}_{\mathsf{ks}} + \sqrt{f_1'}\mathsf{B}_{\mathsf{scale}}$$

Since R_ℓ increases with larger ℓ's, we have to satisfy this inequality for the largest modulus $\ell = L-2$. Moreover, $R_{L-2} > f_1\mathsf{B}_{\mathsf{scale}}$. Since we want that this term is as close to $\mathsf{B}_{\mathsf{scale}}$ as possible, we have to set $f_0 \sim \mathsf{B}_{\mathsf{scale}}/K\mathsf{B}_{\mathsf{ks}}$, for a *large enough* constant $K \in \mathbb{N}$ (i.e., we can take $K \sim 100$). Namely,

$$P \sim \begin{cases} Kq_{L-2}\mathsf{B}_{\mathsf{ks}}/\mathsf{B}_{\mathsf{scale}} & \text{for GHS} \\ K\sqrt{L-1}q_{L-2}\mathsf{B}_{\mathsf{ks}}/\mathsf{B}_{\mathsf{scale}} & \text{for GHS-RNS} \\ K\beta\sqrt{\log_\beta(q_{L-2})}\mathsf{B}_{\mathsf{ks}}/\mathsf{B}_{\mathsf{scale}} & \text{for Hybrid} \\ K\sqrt{\omega(L-1)}p_{L-2}^{L/\omega}\mathsf{B}_{\mathsf{ks}}/\mathsf{B}_{\mathsf{scale}} & \text{for Hybrid-RNS.} \end{cases} \tag{22}$$

Equation (21) becomes

$$\frac{\xi B^2}{p_\ell} + \Big(\frac{2\xi\tau\sqrt{f_1}}{p_\ell}\mathsf{B}_{\mathsf{scale}} - 1\Big)B + \frac{\xi\tau^2 f_1}{p_\ell}\mathsf{B}_{\mathsf{scale}}^2 + \sqrt{f_1'}\mathsf{B}_{\mathsf{scale}} < 0.$$

Thus, to satisfy this inequality (in B), we again must have a positive discriminant, that is, $1 - 4\xi(\sqrt{f_1'} + \tau\sqrt{f_1})\mathsf{B}_{\mathsf{scale}}/p_\ell \geq 0$. We then have

$$p_1 \sim \ldots \sim p_{L-2} \sim 4\xi(\sqrt{f_1'} + \tau\sqrt{f_1})\mathsf{B}_{\mathsf{scale}}. \tag{23}$$

Finally, if we set p_ℓ as in (23), we have the discriminate equal to zero and we can find B with

$$B \sim (2\sqrt{f_1'} + \tau\sqrt{f_1})\mathsf{B}_{\mathsf{scale}}. \tag{24}$$

Now we can compute, the bottom modulus p_0, which is the last modulus in the prime chain, starting from Equation (15). Since P is as in Equation (22), $\tau\nu_{\mathsf{ks}} = \tau(f_0\mathsf{B}_{\mathsf{ks}}/P + \sqrt{f_1}\mathsf{B}_{\mathsf{scale}}) \sim \tau\sqrt{f_1}\mathsf{B}_{\mathsf{scale}}$ and thanks to Equation (24), we have

$$p_0 > 2c_m\xi\big(B + \tau\nu_{\mathsf{ks}}\big)^2 \sim 8c_m\xi\big((\sqrt{f_1'} + \tau\sqrt{f_1})\mathsf{B}_{\mathsf{scale}}\big)^2. \tag{25}$$

Base Model. For the base model with $\tau = 0$, Equations (23) to (25) are as before, instead P as in Equation (22) can be decrease to either

$$Kq_{L-3}\mathsf{B}_{\mathsf{ks}}/\mathsf{B}_{\mathsf{scale}} \text{ in the GHS case}$$

or

$$Kq_{L-3}\sqrt{L-2}\mathsf{B}_{\mathsf{ks}}/\mathsf{B}_{\mathsf{scale}} \text{ for GHS-RNS.}$$

Indeed, R_ℓ changes to either $\frac{q_{\ell-1}}{P}\mathsf{B}_{\mathsf{ks}} + f_1^{ks}\mathsf{B}_{\mathsf{scale}}$ (GHS) or $\frac{q_{\ell-1}\sqrt{L-2}}{P}\mathsf{B}_{\mathsf{ks}} + f_1^{ks}\mathsf{B}_{\mathsf{scale}}$ (GHS-RNS) (see also [15] for the specific case when $\tau = 0$ and $\mathsf{B}_{\mathsf{clean}} = 1$).

The Key Switching Modulus P The biggest modulus for GHS and Hybrid key-switching is $Q = Pq_{L-1}$, where $q_{L-1} = q = p_0 p_{L-1} \prod_{\ell=1}^{L-2} p_\ell$ is the the ciphertext modulus and the value of P is summarized in Table 1.

ks	RNS	Case	P
GHS	–	$\tau \neq 0$	$Kq_{L-2}\frac{\mathsf{B}_{\mathsf{ks}}}{\mathsf{B}_{\mathsf{scale}}}$
		$\tau = 0$	$Kq_{L-3}\frac{\mathsf{B}_{\mathsf{ks}}}{\mathsf{B}_{\mathsf{scale}}}$
	✓	$\tau \neq 0$	$Kq_{L-2}\sqrt{L-1}\frac{\mathsf{B}_{\mathsf{ks}}}{\mathsf{B}_{\mathsf{scale}}}$
		$\tau = 0$	$Kq_{L-3}\sqrt{L-2}\frac{\mathsf{B}_{\mathsf{ks}}}{\mathsf{B}_{\mathsf{scale}}}$
Hybrid	–	*any*	$K\beta\sqrt{\log_\beta(q_{L-2})}\frac{\mathsf{B}_{\mathsf{ks}}}{\mathsf{B}_{\mathsf{scale}}}$
	✓	*any*	$K\sqrt{w(L-1)}p_{L-2}^{L/\omega}\frac{\mathsf{B}_{\mathsf{ks}}}{\mathsf{B}_{\mathsf{scale}}}$

Table 1. The key switching modulus P.

From Table 1, one can also see that the hybrid key switching provide smaller P, where $K \in \mathbb{N}$ is a *large enough* constant (i.e., we can take $K \sim 100$), $\mathsf{B}_{\mathsf{const}}$, $\mathsf{B}_{\mathsf{clean}}$, $\mathsf{B}_{\mathsf{scale}}$ and B_{ks} as in Equations (6) to (8) and (10), respectively.

Modulus Size for Power-of-Two Cyclotomics In Tables 2 and 3, we summarize the moduli size considering $\Phi_m(x) = x^n + 1$ and $n = 2^\kappa$ (and so $m = 2n$ and $\phi(m) = n$) in the case of Base Model and Model 2 depicted in Figure 1, respectively. We denoted by G/H, the GHS, and the Hybrid key-switching variant without RNS, respectively. It is worth noting that in the case of G/H-RNS, we approximate $\tau\sqrt{k} + 2\sqrt{k+1} - 1 \sim \sqrt{k}(\tau + 2)$.

p_0	p_ℓ	p_{L-1}	
$8k\xi\mathsf{B}_{\mathsf{scale}}^2$	$4\sqrt{k}\xi\mathsf{B}_{\mathsf{scale}}$	$\mathsf{B}_{\mathsf{clean}}/(2\sqrt{k}-1)\mathsf{B}_{\mathsf{scale}}$	G/H-RNS
$8\xi\mathsf{B}_{\mathsf{scale}}^2$	$4\xi\mathsf{B}_{\mathsf{scale}}$	$\mathsf{B}_{\mathsf{clean}}/\mathsf{B}_{\mathsf{scale}}$	else

Table 2. Base Model, namely, case with $\tau = 0$.

p_0	$32\eta^2\mathsf{B}_{\mathsf{const}}^2\left(3\tau\beta\sqrt{(L-1)\log_2(t\phi(m))}\mathsf{B}_{\mathsf{ks}}+\mathsf{B}_{\mathsf{scale}}\right)^2$	BV
	$8\left(\eta(\tau+1)\mathsf{B}_{\mathsf{const}}\mathsf{B}_{\mathsf{scale}}\right)^2$	G/H
	$8k\left(\eta(\tau+1)\mathsf{B}_{\mathsf{const}}\mathsf{B}_{\mathsf{scale}}\right)^2$	G/H-RNS
p_ℓ	$4\eta^2\mathsf{B}_{\mathsf{const}}^2\left(3\tau\beta\sqrt{(L-1)\log_2(t\phi(m))}\mathsf{B}_{\mathsf{ks}}+\mathsf{B}_{\mathsf{scale}}\right)$	BV
	$4\eta^2(\tau+1)\mathsf{B}_{\mathsf{const}}^2\mathsf{B}_{\mathsf{scale}}$	G/H
	$4\eta^2\sqrt{k}(\tau+1)\mathsf{B}_{\mathsf{const}}^2\mathsf{B}_{\mathsf{scale}}$	G/H-RNS
p_{L-1}	$\mathsf{B}_{\mathsf{clean}}/\left(6\tau\beta\sqrt{(L-1)\log_2(t\phi(m))}\mathsf{B}_{\mathsf{ks}}+\mathsf{B}_{\mathsf{scale}}\right)$	BV
	$\mathsf{B}_{\mathsf{clean}}/(\tau+1)\mathsf{B}_{\mathsf{scale}}$	G/H
	$\mathsf{B}_{\mathsf{clean}}/\sqrt{k}(\tau+2)\mathsf{B}_{\mathsf{scale}}$	G/H-RNS

Table 3. Modului size for power-of-two polynomial in the case of Model 2.

3.6 Parameters Specification

In Table 4 we summarize all the parameters specification, where, we recall that $q_\ell = \prod_{j=0}^{\ell} p_j$ for any level $0 \leq \ell \leq L-1$ and, for the RNS variant of GHS and Hybrid, we have $P = \prod_{j=1}^{k} P_j$.

$t \equiv 1 \mod m$	for CRT
$\gcd(t, q) = 1$	for security reason
P_i and p_j small primes	for RNS
$p_j \equiv 1 \mod m$ and $P_i \equiv 1 \mod m$	for efficient NTT
p_ℓ roughly of the same size	with $1 \leq \ell \leq L-2$

Table 4. Required parameter specification

Moreover, for the scaling procedure (see Section 2.5), one can choose

$$p_i \equiv 1 \mod t \quad \text{and} \quad P \equiv 1 \mod t.$$

3.7 Security Analysis

In previous works, Costache and Smart [9], following Gentry *et al.* [15] analysis, used the security formula by Lindner-Peikert [20], that is $\log t_{BKZ}(\delta_0) = 1.8/\log\delta_0 - 110$. Lindner and Peikert's estimation has a few inaccuracies [2] and turns out to be too optimistic. Thus, Costache *et al.* [10] propose parameters according to the Homomorphic Encryption Standard [4] for a uniformly distributed ternary secret and $\lambda = 128$.

To enable a more flexible and faster parameter selection, we propose an empirically derived formula linking the security level λ with the dimension n for a given ciphertext modulus size ($\log q$). This enables a fast security estimate for parameter generation. Let us consider a full-rank lattice $\mathcal{L}$. We know that the shortest vector of $\mathcal{L}$ has norm $||\mathbf{b}_1|| = \delta_0^k q^{n/k}$ (see Equation (4)). To perform

lattice reduction on $\mathcal{L}$, the LWE attacker has to choose the number of samples M, namely the subdimension, such that $||\mathbf{b}_1|| = \delta_0^M q^{n/M}$ is minimized.

Micciancio and Regev [24] showed that this minimum is obtained when $M = \sqrt{n \log q / \log \delta_0}$. We can suppose that we should reduce the basis enough so that $||\mathbf{b}_1|| = q$. This implies that $\log q = \log(\delta_0^M q^{n/M}) = 2\sqrt{n \log q \log \delta_0}$, that is

$$n = \log q / (4 \log \delta_0). \tag{26}$$

Substituting Equation (26) in Equation (5), we have a bound linking λ, n. Then, we do the following:

1. We run the Lattice Estimator [5] for the dimensions $n = 2^k$ as well as the secret distribution $\chi_s = \mathcal{U}_3$ and $\chi_s = \chi_e$. We choose $k \in \{10, \ldots, 15\}$ following the Homomorphic Encryption Standard [3]. Generating this necessary data, especially for higher degrees, is computationally intensive. For $\log q = 600$ and $n = 2^{15}$ for example, it takes roughly 2.5h to evaluate the security for three attacks.
2. Starting from the theoretical bound linking λ, n, and q, we find a parameterized function that follows the data points generated with the lattice estimator.
3. Finally, we model the resulting formula with coupled optimization to find the best constants. To ensure that our model provides accurate estimations of security levels, we place constraints on the selection of constants. Specifically, we instructed the model to choose constants such that the resulting output slightly underestimates the security level rather than overestimating it. This ensures that we maintain a high degree of confidence in the estimated model (see Table 6) while avoiding potential false assurances that could have resulted from overestimation.

Using this process, the resulting function is

$$\lambda \approx -\log\left(\frac{\mathrm{A}\log q}{n}\right)\frac{\mathrm{B}n}{\log q} + \mathrm{C}\left(\frac{\log q}{n}\right)^D \log\left(\frac{n}{\log q}\right) \tag{27}$$

with constants

$$\begin{aligned} &\mathrm{A} = 0.05 \;\; \mathrm{B} = 0.33 \;\; \mathrm{C} = 17.88 \;\; \mathrm{D} = 0.65 \;\; \text{if } \chi_s = \mathcal{U}_3 \text{ and} \\ &\mathrm{A} = 3.87 \;\; \mathrm{B} = 0.74 \;\; \mathrm{C} = 12.72 \;\; \mathrm{D} = 0.17 \;\; \text{if } \chi_s = \chi_e. \end{aligned}$$

3.8 A Parameter Generator for BGV

The parameter generator for BGV provides an accessible way to our theoretical work. Most importantly, developers can use the generator and receive a simple code example for state-of-the-art libraries. The generator itself is written in Python and it is publicly available on GitHub[4]. On request, we can also disclose the code to the reviewers.

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

Supporting Arbitrary Circuit Models. Although we only theoretically analyze some circuit models in our work, the generator itself is easily extendable to arbitrary circuit models that an advanced user can compose by themself. This, however, requires a simplification of the key switching bound for each variant, as we otherwise have circular dependencies on the size of the moduli in the prime chain.

We take a straightforward and practical-oriented approach: We assume a bound on $p_\ell \leq 2^b$, per default $2^b = 2^{128}$, and extrapolate the bound to q_ℓ as $L2^b$. For the BV, BV-RNS, and Hybrid variants, we fix β and ω, per default as $\beta = 2^{10}$ and $\omega = 3$. For the extension modulus P, we choose a constant K, per default $K = 100$. Then, we set $P = KL2^b$ for the GHS and GHS-RNS variants, $P = K\beta\sqrt{\log_\beta L2^b}$ for the Hybrid variant and $P = K\sqrt{\omega L}(2^b)^{\lceil L/\omega \rceil}$ for the Hybrid-RNS variant. We can now easily compose different circuit models using these constant bounds and solve the inequalities for B programmatically by finding the local minimum in the interval $[0, 2^b]$.

Using the Parameter Generator. An interactive mode prompts the user with several questions for required and optional inputs. We list the required inputs in the first part and optional inputs in the second part of Table 5. After providing all the required information, the user receives the output in text form and, if the library option is chosen, the generated code. The output for the ciphertext modulus contains the bound on the ciphertext modulus itself as well as the bounds for the bottom, middle and top modulus, respectively. The generated code is a setup routine for the chosen library and provides references to further code examples within the respective library.

Model	`'Base'`, `'Model1'`, `'Model2'`, `'OpenFHE'`
t or $\log t$	any integer ≥ 2
λ or m	any integer ≥ 40 or ≥ 4, respectively
M, η	any integer > 0
τ	any integer ≥ 0
Library	`'None'`, `'OpenFHE'`, `'PALISADE'`, `'SEAL'`
Full Batching	full batching with t, `'True'` or `'False'`
Secret Distribution	`'Ternary'`, `'Error'`
Key Switching	`'Hybrid'`, `'BV'`, `'GHS'`
β	any integer ≥ 2
ω	any integer ≥ 1

Table 5. Required and optional inputs to the parameter generator

Limitations. Due to the internal workings of the libraries, we cannot guarantee that all parameter sets work. For example, OpenFHE supports only a plaintext

modulus of up to 60 bits. Thus, choosing a larger plaintext modulus will result in non-working code. However, we are happy to work with the community to integrate checks on these constraints as users encounter them.

4 Results

In the following, we compare our security formula as well as our bounds theoretically and practically to previous work.

4.1 Security Parameter Evaluation

In Table 6, we compare the security levels provided by our formula with the Lattice Estimator and the LP model. For the LP model, we use $\lambda \leq 7.2n/\log q - 78.9$, substituting Equation (26) in the LP formula, where the run-time is expressed in units of computation rather than seconds. Our formula yields estimations very close to those computed using the Lattice Estimator, whereas those computed using the LP method tend to overestimate the security level in almost every case. It is worth noting, however, that as shown in Table 6, the accuracy of our estimation decreases slightly as n increases.

In Figure 2, we provide a visualization of the formula together with the original data points by the Lattice Estimator for the secret distribution $\mathcal{U}_3$.

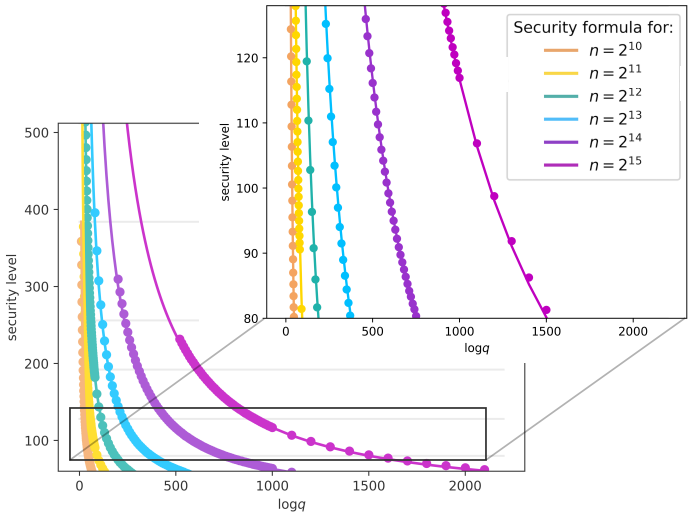

Fig. 2. The security formula with data points of the Lattice Estimator for $\chi_s = \mathcal{U}_3$.

Note that for the GHS and Hybrid key switching variants, we have to consider the bigger modulus size $\log Q = \log qP$ to evaluate security.

$n = 2^{10}$				$n = 2^{13}$			
$\log q$	Estimator	Our Work	LP	$\log q$	Estimator	Our Work	LP
16	221	222	382	114	258	252	438
18	195	195	331	147	193	192	322
19	184	184	244	148	192	190	320
25	137	137	216	210	131	131	202
26	132	132	216	214	128	128	197
27	127	127	216	215	127	127	195
30	114	114	167	270	101	101	139
35	97	97	132	300	91	91	118
42	80	81	97	340	80	80	95
43	79	79	93	344	79	79	93
$n = 2^{11}$				$n = 2^{14}$			
28	258	257	447	230	257	250	434
29	248	247	429	231	255	248	432
35	202	201	342	293	195	192	324
36	196	195	331	297	192	189	318
37	190	190	319	298	191	188	317
53	129	129	199	400	138	138	216
54	126	127	194	429	128	128	196
85	80	80	95	430	128	127	195
86	79	79	93	680	81	80	94
$n = 2^{12}$				$n = 2^{15}$			
56	261	257	448	400	303	291	511
57	256	252	438	460	258	250	434
58	251	247	430	586	196	192	324
73	194	193	325	598	192	188	317
74	191	190	320	850	131	129	199
107	128	128	197	859	129	128	196
108	126	127	194	870	128	126	192
170	80	80	95	871	127	126	192
171	79	80	93	1300	86	84	103
172	79	79	93	1400	80	78	89

Table 6. Comparison between the security level provided by our formula, the Lattice estimator, and the LP model with secret distribution $\mathcal{U}_3$.

4.2 Parameter Generation

We now compare our bounds theoretically with Kim et al. [19], who use the infinity norm, and practically with the implementation of their theoretical work in OpenFHE [6].

Comparing our bounds theoretically, we use the ring expansion factor $\delta_{\mathcal{R}} = \phi(m)$ for the infinity norm. Here, our results are up to 20 % better for the bottom modulus and up to 42 % better for the middle modulus (see also Figure 3), reducing the size of the full ciphertext modulus significantly. Our better results for the top modulus on the other hand do not have any effect in practice as for NTT compatibility, we have to choose each modulus $p > m$ anyway.

For an actual application of the bounds, Kim et al. [19] suggest using the expansion factor $\delta_{\mathcal{R}} = 2\sqrt{\phi(m)}$ instead of $\delta_{\mathcal{R}} = \phi(m)$ as the probability of the latter is exponentially low and the former is closer to observations in practice [16]. The parameter generation in OpenFHE also uses this heuristic expansion factor. Interestingly enough, this generates moduli sizes that are very similar compared to our bounds based on the purely theoretical approach with the canonical embedding norm. However, in the middle moduli, which for increasing multiplicative depth is dominating the overall modulus size, we still perform better than the heuristic bound with the infinity norm.

For the security level $\lambda = 128$, the plaintext modulus $t = 2^{16} + 1$, secret key distribution $\chi_s = \mathcal{U}_3$, and $D = 6$, we map our results in the OpenFHE model without any rotations, that is $\tau = 0$, and compare them with the results of OpenFHE theoretically in Figure 3, that is with $\delta_{\mathcal{R}} = d$, and practically in Figure 4, that is for $\delta_{\mathcal{R}} = 2\sqrt{d}$. For a fair comparison of p_0, we used $p_0 p_1$ for OpenFHE as their model does not allow any operations in the last level compared to our bottom modulus.

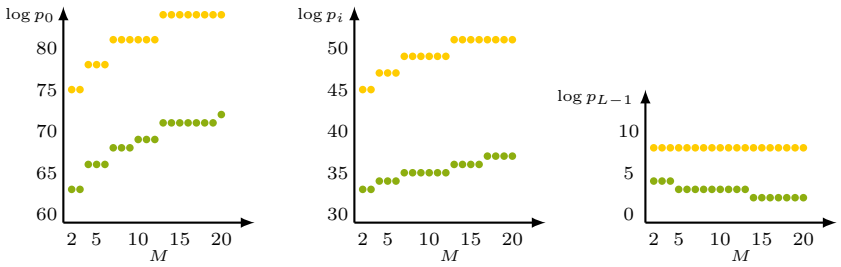

Fig. 3. Comparison of theoretical prime sizes across multiplicative depths M with $\lambda = 128$ and $t = 2^{16} + 1$ for OpenFHE ● and our ● parameter generation.

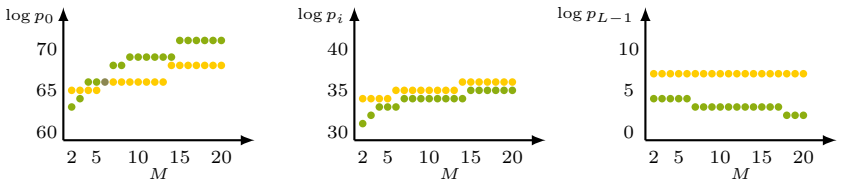

Fig. 4. Comparison of practical prime sizes across multiplicative depths M with $\lambda = 128$ and $t = 2^{16} + 1$ for OpenFHE ● and our ● parameter generation (● is matching).

5 Conclusion

Choosing parameters such as the polynomial degree or the ciphertext modulus is challenging for BGV and requires an in-depth analysis of the noise growth for each operation. Use-case-specific aspects, such as the order of operations, impact noise growth, further increasing the challenge with parameter generation. Additionally, the generated parameter sets have to be secure with respect to the underlying RLWE assumption, requiring good lower bounds on the ciphertext modulus.

In this work, we improve the parameter generation across all steps of its process. First, we extend previous analyses bringing together the DCRT representation and the canonical embedding norm and improving upon the existing state-of-the-art. Using these bounds, we proposed new circuit models, including essential BGV operations such as constant multiplication or rotations. Additionally, we provide an empirically derived, closed formula to estimate the security for a given parameter set based on coupled optimization for different secret key distributions. Finally, we combine our theoretical research and implement our results in an interactive parameter generator for BGV, which outputs easy-to-use code snippets for the state-of-the-art libraries OpenFHE, PALISADE, and SEAL.

Acknowledgements We want to thank Anna Hambitzer for her helpful comments on coupled optimization.

Bibliography

[1] 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] 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, *Advances in Cryptology – EUROCRYPT 2017*, pages 103–129, Cham, 2017. Springer International Publishing.

[3] 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.

[4] Martin R Albrecht, Carlos Cid, Jean-Charles Faugere, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the bkw algorithm on lwe. *Designs, Codes and Cryptography*, 74(2):325–354, 2015.

[5] 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.

[6] 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. `https://eprint.iacr.org/2022/915`.

[7] Zvika Brakerski and Vinod Vaikuntanathan. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 505–524, Berlin, Heidelberg, 2011. Springer.

[8] Hao Chen, Miran Kim, Ilya Razenshteyn, Dragos Rotaru, Yongsoo Song, and Sameer Wagh. Maliciously secure matrix multiplication with applications to private deep learning. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 31–59. Springer, 2020.

[9] Ana 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.

[10] Anamaria Costache, Kim Laine, and Rachel Player. Evaluating the effectiveness of heuristic worst-case noise analysis in FHE. In *European Symposium on Research in Computer Security*, pages 546–565. Springer, 2020.

[11] Anamaria Costache, Lea Nürnberger, and Rachel Player. Optimizations and trade-offs for helib. *Cryptology ePrint Archive*, 2023.
[12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pages 643–662. Springer, 2012.
[13] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, pages 31–51, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
[14] Craig Gentry. *A fully homomorphic encryption scheme*, volume 20. Stanford university Stanford, 2009.
[15] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic Evaluation of the AES Circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 850–867, Berlin, Heidelberg, 2012. Springer.
[16] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An improved RNS variant of the BFV homomorphic encryption scheme. In *Cryptographers' Track at the RSA Conference*, pages 83–105. Springer, 2019.
[17] Shai Halevi and Victor Shoup. Design and implementation of helib: a homomorphic encryption library. *Cryptology ePrint Archive*, 2020.
[18] Ilia Iliashenko. Optimisations of fully homomorphic encryption, 2019.
[19] Andrey Kim, Yuriy Polyakov, and Vincent Zucca. Revisiting homomorphic encryption schemes for finite fields, 2021.
[20] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *Cryptographers' Track at the RSA Conference*, pages 319–339. Springer, 2011.
[21] 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, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
[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.
[23] 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.
[24] Daniele Micciancio and Oded Regev. *Lattice-based Cryptography*. Springer, Berlin, Heidelberg, 2009.
[25] PALISADE. `https://palisade-crypto.org`.
[26] 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.
[27] 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.
[28] Gregor Seiler. Faster avx2 optimized ntt multiplication for ring-lwe lattice cryptography, January 2018.