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. The Brakerski Gentry Vaikuntanathan (BGV) scheme is a state-of-the-art fully homomorphic encryption (FHE) scheme. Encryption is based on the Learning with Errors over rings (RLWE) assumption and thus, each ciphertext has an associated error that grows with each homomorphic operation. To avoid failure during decryption, the growing error needs to stay below a certain threshold. This requires a trade-off between security and error margin that influences the parameters specific to each use case. Choosing such parameters, for example the polynomial degree or the ciphertext modulus, is a challenge and requires expert knowledge specific to each scheme.

In this work, we improve the current state-of-the-art of parameter generation. We provide a comprehensive analysis of BGV parameters for the Double Chinese Remainder Theorem (DCRT) representation, study the error bounds for additional cases such as rotations or constant multiplications and provide improved noise estimates for parameter generation. We also empirically derive a closed formula enabling fast security estimates given a ciphertext modulus and polynomial degree. Finally, we combine our theoretical research and provide an interactive parameter generator for the leveled BGV scheme which output easy-to-use code snippets for PALISADE, an open-source FHE library.

Keywords: Fully Homomorphic Encryption, BGV, Parameter Generation, PALISADE

1 Introduction

Since Gentry's seminal work in 2009 [10], fully homomorphic encryption (FHE) has attracted a lot of attention from the cryptographic research community [1, 15, 16]. FHE enables arbitrary computations on encrypted data and 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 to leak any sensitive data.

The BGV scheme [4] is currently considered to be one of the state-of-the-art FHE schemes. BGV is based on the Learning with Errors (LWE) problem [19] and its ring variant RLWE [14]. 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 (big ciphertext modulus).

In general, choosing parameters such as the polynomial degree d or the ciphertext modulus q is a challenge with FHE schemes and requires expert knowledge specific to each scheme. There are multiple parameters to consider and users and developers alike have to make many choices with state-of-the-art software libraries. A real-world example is the programming interface of PALISADE [18], an open-source FHE library that also implements BGV. A user needs to choose a polynomial implementation as well as seven additional parameters (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

For researchers familiar with FHE, this flexibility is valuable. For other users however, this burden of choice increases the difficulty of simply using the library securely.

The FHE community took a first step towards standardization with the Homomorphic Encryption Security Standard [2]. The standard provides parameter sets based on the LWE Estimator [3], 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 use these look-up tables in order to simplify the parameter selection for users. As FHE parameters depend on the specific use case, the trade-off is a non-optimized parameter generation leading to larger than necessary parameters. This in turn negatively affects the runtime and memory usage of FHE implementations. Thus, generating high-quality parameters is important for FHE in general.

Related Work For BGV, there currently are two main approaches analyzing the error growth: with the canonical norm [6, 7, 12] or with the infinity norm [13]. While the canonical norm is known to result in better parameters, only the last work analyzed the BGV operations for the DCRT representation, the best known approach to implementing RLWE-based schemes. Additionally, while there are case studies with specific parameter sets [5], the circuit models for parameter generation have remained rather simple and static [11].

The main idea of our work is to improve the current state-of-the-art of parameter generation and the usability of the BGV scheme with the following contributions:

- We provide a comprehensive analysis of BGV parameters for the DCRT representation. We study the bounds for the error growth considering additional cases such as rotations and constant multiplications and provide improved noise estimates for parameter generation (Section 3).
- We provide an interactive parameter generator for the leveled BGV scheme using our theoretical and empirical formulas (Section 4). The generator outputs easy-to-use code snippets with example circuits for PALISADE as well as benchmarking code for the specific parameter set.
- We empirically derive a closed formula enabling fast estimation of the security of a parameter set. More specifically, we output a security estimate for a given ciphertext modulus and polynomial degree (Section 4.1).
- Moreover, in Section 2.5, we also provide a comprehensive manual on the bounds for the error growth and the parameters used in the leveled BGV scheme, following and collecting previous studies [2, 6, 7, 11, 12, 13].

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 are roughly of the same size and $p_i \equiv 1 \mod m$ [11]. 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 mod m.

We denoted by χ_e the RLWE error distribution, typically a discrete Gaussian with standard deviation $\sigma = 3.19$ [2], 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 canonical norm In this section we recall the result of [6, 7, 12]. Let $a \in R$ be a polynomial. 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 [8].

Let us consider a random $a \in R$ where each coefficient is sampled independently from on 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 [6]. 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 randomly and γ is a constant, the following holds for the variances [7]:

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

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 [6], 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 [15]. In [3], 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 [20]. In these algorithms, the time complexity and the outcome quality (i.e. the orthogonality of the reduced basis) is characterised by the Hermite factor [9]. Specifically, the run time of the BKZ algorithm is higher when the root-Hermite factor δ_0 is smaller [20]. This result is also supported by a realistic estimation provided in [3], 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 $\mathbb{Z}_{q_i}$ for $q = \prod q_i$. For pairwise coprime q_i, we define a ring isomorphism $\mathbb{Z}_q \cong \mathbb{Z}_{q_1} \times \ldots \times \mathbb{Z}_{q_k}$ with

$$x \bmod q \mapsto (x \bmod q_1, \ldots, x \bmod q_k).$$

In the context of BGV, we decompose a polynomial in $\mathcal{R}_q$ into smaller polynomials in $\mathcal{R}_{q_1} \times \ldots \times \mathcal{R}_{q_k}$.

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 [21].

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 [11]. 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 [7, 12]. 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

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

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

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

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

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

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 [6].

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:

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

The critical quantity is thus defined as $[c_0 + c_1 \cdot s]_{q_\ell}$ for the associated level ℓ.

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

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

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

$$\mathsf{B}_{\mathsf{scale}} = Dt\sqrt{\frac{\phi(m)}{12}(1 + \phi(m)V_s)}. \tag{7}$$

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

For the RNS implementation, we have to apply a *fast base extension* to $\boldsymbol{\delta}$ from $\frac{q_{\ell'}}{q_\ell}$ to $q_{\ell'}$. For two moduli $q = \prod_{i=1}^{k} q_i$ and $p = \prod_{j=1}^{k'} p_j$ and a polynomial α, we define it as

$$\mathrm{BaseExt}(\alpha, q, p) = \sum_{i=1}^{k} \left[\alpha \frac{q_i}{q}\right]_{q_i} \frac{q}{q_i} \bmod p_j.$$

This outputs $\alpha + qu$ in the RNS base p for an error polynomial u. In our case, we have to set

$$\boldsymbol{\delta} = t\,\mathrm{BaseExt}(-\mathbf{c}t^{-1}, \frac{q_\ell}{q_{\ell'}}, q_{\ell'}).$$

Note that for $\ell' = \ell - 1$, $\text{BaseExt}(-\mathbf{c}t^{-1}, p_\ell, q_{\ell'}) = [-\mathbf{c}t^{-1}]_{q_\ell/q_{\ell'}}$. We analyze the impact on 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. In this work, we consider 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 ω to reduce the error growth [4]. For polynomials α and β and $l = \lfloor \log_\omega q_\ell \rfloor + 1$, we define

$$\mathcal{D}_\omega(\alpha) = ([\alpha]_\omega, [\lfloor \alpha/\omega \rfloor]_\omega, \ldots, [\lfloor \alpha/\omega^{l-1} \rfloor]_\omega)$$

$$\mathcal{P}_\omega(\beta) = ([\beta]_{q_\ell}, [\beta\omega]_{q_\ell}, \ldots, [\beta\omega^{l-1}]_{q_\ell}).$$

It follows that, for any $\alpha, \beta \in \mathcal{R}_{q_\ell}$, we have $\langle \mathcal{D}_\omega(\alpha), \mathcal{P}_\omega(\beta) \rangle \equiv \alpha \cdot \beta \bmod q_\ell$[13].

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

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

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

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

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

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,

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

that is,

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

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

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

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

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

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

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

where

$$\mathsf{B}_{\mathsf{ks}} = Dt\phi(m)\sqrt{V_e/12}. \tag{8}$$

BV-RNS Variant For the BV-RNS variant, we define $\mathcal{D}$ and $\mathcal{P}$ not with respect to some digit decomposition ω, but rather use the already existing RNS decomposition.

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

We analyze the growth of noise in Section 3.

GHS Variant The GHS variant [11] 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 (7) and (8), 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 apply the same adaptations to $\boldsymbol{\delta}$ as to the RNS version of modulus switching. Additionally, we have to apply the fast base extension to c_2 and extend it with

$$\text{BaseExt}(c_2, q_\ell, Q_\ell) = c_2 + q_\ell u.$$

We analyze the noise growth in Section 3.

Hybrid Variant The Hybrid variant combines the BV and GHS variants [11]. 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 $\mathbf{a} \leftarrow \mathcal{U}_{Q_L}^l$, $\mathbf{e} \leftarrow \chi_e^l$.
Output key switching key

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

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

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

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 $\nu' = \nu P + \omega l \mathsf{B}_{\mathsf{ks}}$, where B_{ks} is as an Equation (8). Thus, the error after the modulus switching procedure is

bounded by $\frac{q_\ell}{Q_\ell}||\nu'||^{can} + \mathsf{B}_{\mathsf{scale}}$, that is,

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

where $\mathsf{B}_{\mathsf{scale}}$ is defined as Equation (7).

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 chuncks of size $l = \frac{L}{\omega} - 1$. Hence, we set $\tilde{q}_i = \prod_{j=il}^{il+l-1} p_j$ and define $\mathcal{D}$ and $\mathcal{P}$ as

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

Also, we now have to base extend from each $\tilde{q}_i$ to Q_ℓ instead.

3 Analyzing Error and Modulus Bounds

In this section, we study the modulus size p_ℓ for any level ℓ. Our approach is extends previous work [6, 7, 11] by using new bounds for the DCRT representation, including essential BGV operations such as rotations and constant multiplications and allows for multiple key switching approaches.

We start by analysing the error bounds for modulus and key switching in the DCRT representation. Then, we follow up by specifying our circuit model and finish our analysis by taking a look at the decryption modulus q_0, in the following called bottom modulus, the top modulus q_{L-1} and the remaining middle moduli q_ℓ.

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 (7).

When scaling by $k > 1$ moduli however, we have to adjust our bound due to the base extension. 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

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

Using the fast base extension, we have $\boldsymbol{\delta} = t\,\mathrm{BaseExt}(-\mathbf{c}t^{-1}, \frac{q_\ell}{q_{\ell'}}, q_{\ell'})$. The variance follows as

$$V_{\delta_i} = \left(t\sqrt{k}\frac{q_\ell}{q_{\ell'}p_\ell}\frac{p_\ell}{\sqrt{12}}\right)^2 = \left(t\sqrt{\frac{k}{12}}\frac{q_\ell}{q_{\ell'}}\right)^2$$

introducing a factor of $\sqrt{k}$ and thus $||\nu_{\mathsf{ms}}||^{can} \leq \frac{q_{\ell'}}{q_\ell}||\nu||^{can} + \sqrt{k}\mathsf{B}_{\mathsf{scale}}$.

3.2 Key Switching

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

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

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

$$\begin{aligned}c_0' + c_1's &\equiv Pc_0 + (c_2 + q_\ell u)\cdot\mathsf{ks}_0 + (Pc_1 + (c_2 + q_\ell u)\cdot\mathsf{ks}_1)s &&(\mathrm{mod}\ Q_\ell)\\ &\equiv P[c_0 + c_1s + c_2s^2]_{q_\ell} + t(c_2 + q_\ell u)e &&(\mathrm{mod}\ Q_\ell)\end{aligned}$$

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

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

For the Hybrid-RNS variant, we can combine our previous analyses. However, we again have to adjust for the fact that 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

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

3.3 Modeling the Homomorphic Circuit

In general, we split the homomorphic circuit in 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 do 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 modulus switching procedure. This reduce 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 such that decryption can be performed correctly.

At each level (except the first one), we operate on $\eta+1$ ciphertexts c_i in parallel. Then, one constant multiplication can be performed followed up by τ rotations. Finally, the ciphertexts are added up and used as input to one multiplication before modulus switching is applied. Note that within that model, all parameters can be chosen as required by the use case.

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.

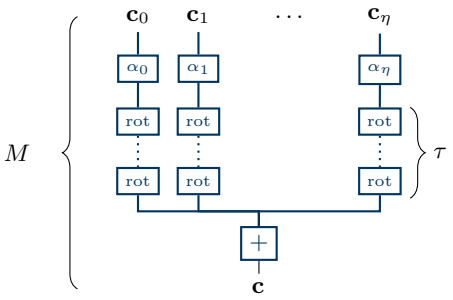

Fig. 1. Our analysis model depicted as circuit.

3.4 Error Analysis

Let us consider the multiplication of two ciphertexts, $\mathbf{c} \cdot \mathbf{c}'$, both output of the circuit in Figure 1. Each of $\mathbf{c}_i$ ciphertext has starting noise B. After performing a constant multiplication the error grows from B to $B\mathsf{B}_{\mathsf{const}}$, where $\mathsf{B}_{\mathsf{const}} = Dt\sqrt{\phi(m)/12}$ (as in Section 2.5). Then we apply τ rotations. Note that, the rotations itself do not influence the noise directly, however the key switching back to the original key adds key switching noise v_{ks}, which depend on the key

switching method:

$$v_{ks} = \begin{cases} \omega\sqrt{\log_\omega(q_\ell)}\mathsf{B_{ks}} & \text{(BV)} \\ p_\ell\sqrt{\ell+1}\mathsf{B_{ks}} & \text{(BV - RNS)} \\ f_0^{ks}\mathsf{B_{ks}}/P + f_1^{ks}\mathsf{B_{scale}} & (others) \end{cases} \tag{9}$$

where

$$f^{ks} = \begin{cases} (q_\ell, 1) & \text{(GHS)} \\ (q_\ell\sqrt{\ell+1}, \sqrt{k}) & \text{(GHS - RNS)} \\ (\omega\sqrt{\lfloor\log_\omega(q_\ell)\rfloor}, 1) & \text{(Hybrid)} \\ (p_\ell^{L/\omega}\sqrt{w(\ell+1)}, \sqrt{k}) & \text{(Hybrid - RNS)} \end{cases} \tag{10}$$

So, after the τ rotations the bounded error of each ciphertext $\mathbf{c}_i$ grows from $B\mathsf{B_{const}}$ to $B\mathsf{B_{const}} + \tau v_{ks}$. Thus we sum all the resulting ciphertexts together obtaining the final ciphertext $\mathbf{c}$ (with error bounded by $\eta(B\mathsf{B_{const}}+\tau v_{ks})$). Lastly, we multiply $\mathbf{c}$ and $\mathbf{c}'$, both output of the circuit in Figure 1 and the final noise is grew up to

$$(\eta B\mathsf{B_{const}} + \eta\tau v_{ks})^2 = \varepsilon^2 B^2 + \eta^2\tau^2 v_{ks}^2 + 2\eta\tau\varepsilon v_{ks} B,$$

where $\varepsilon = \eta\mathsf{B_{const}}$ and let $\xi = \varepsilon^2$.

In the base model without rotations, the noise magnitude grows from B to ξB^2. Following the same argument as before, we provide several values for ξ which depend on the order of multiplications and additions (see Table 1). If we do not need any constant multiplication, we set $\mathsf{B_{const}} = 1$.

Order	Count #1	Count #2	ξ
Add then Mul	η	1	$\eta^2\mathsf{B}_{\mathsf{const}}^2$
Add then Mul	η, η'	1	$\eta\eta'\mathsf{B}_{\mathsf{const}}^2$
Mul then Add	1	η	$\eta\mathsf{B}_{\mathsf{const}}^2$

Table 1. Defining ξ based on the order of addition and multiplication.

3.5 The Top Modulus

The top modulus is the first modulus in the prime chain, that is p_{L-1}. Before do any operation, we reduce the error $\mathsf{B_{clean}}$ down to B. We use the modulus switching reducing the error and go down to next level:

$$\frac{\mathsf{B_{clean}}}{p_{L-1}} + \mathsf{B_{scale}} < B \iff p_{L-1} > \frac{\mathsf{B_{clean}}}{B - \mathsf{B_{scale}}} \tag{11}$$

Note that, in same specific cases we have that $B - \mathsf{B_{scale}} > \mathsf{B_{clean}}$. So in these cases we do not have any restriction on p_{L-1}.

3.6 Middle Moduli (when $L \geq 3$)

As described before, considering the multiplication of two ciphertexts, $\mathbf{c}$ and $\mathbf{c}'$, output of the circuit in Figure 1, the noise magnitude grows from B to $(\varepsilon B + \eta\tau v_{ks})^2$. Since $\mathbf{c} \cdot \mathbf{c}'$ is a polynomial of degree 2, we have to perform the key switching technique and, to reduce the noise magnitude down to B, we also need the modulus switching procedure.

In the case of GHS and Hybrid key switching, we can actually 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 $q_{\ell-1}/Q_\ell = 1/(Pp_\ell)$. Note that, performing these procedure, we also add the key switching noise v_{ks} (see (9)) and the modulus switching noise $\mathsf{B}_{\mathsf{scale}}$ (as in Equation (7)).

BV key-switching. Since we want to reduce the noise size back to B, after the key switching we have to apply the modulus switching. Thus, we have to set

$$\frac{(\varepsilon B + \eta\tau\omega\sqrt{\lfloor \log_\omega(q_\ell)\rfloor}\mathsf{B}_{\mathsf{ks}})^2}{p_\ell} + \frac{\omega\mathsf{B}_{\mathsf{ks}}\sqrt{\lfloor \log_\omega(q_\ell)\rfloor}}{p_\ell} + \mathsf{B}_{\mathsf{scale}} < B. \tag{12}$$

Note that $\sqrt{\lfloor \log_\omega(q_\ell)\rfloor} \sim \sqrt{(\ell+1)\log_\omega(p_\ell)}$ and, for p_ℓ enough big, we have that $\omega\mathsf{B}_{\mathsf{ks}}\sqrt{\lfloor \log_\omega(q_\ell)\rfloor}/p_\ell \sim 0$. Then,

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

Since Equation (12) must have a positive discriminant, we have

$$1 - \frac{4\eta\tau\varepsilon\omega\sqrt{(\ell+1)\log_\omega(p_\ell)}\mathsf{B}_{\mathsf{ks}}}{p_\ell} - \frac{\xi\mathsf{B}_{\mathsf{scale}}}{p_\ell} \geq 0.$$

Moreover, the p_ℓ's have roughly the same size and we have to satisfy Equation (12) for the largest modulus $\ell = L-2$, thus we can set

$$p_1 \sim \ldots \sim p_{L-2} \sim \eta\varepsilon(4\tau\omega\sqrt{L-1}\log_\omega(\mathsf{B}_{\mathsf{ks}})\mathsf{B}_{\mathsf{ks}} + \mathsf{B}_{\mathsf{const}}\mathsf{B}_{\mathsf{scale}}) \tag{13}$$

So we have $2\eta\tau\varepsilon\omega\sqrt{(\ell+1)\log_\omega(p_\ell)}\mathsf{B}_{\mathsf{ks}}/p_\ell \sim 0$ and the discriminant ~ 0. Thus, $B \sim \frac{1}{2\xi/p_\ell} \sim 2\tau\omega\sqrt{L-1}\log_\omega(\mathsf{B}_{\mathsf{ks}})\mathsf{B}_{\mathsf{ks}}/\mathsf{B}_{\mathsf{const}} + \mathsf{B}_{\mathsf{scale}}/2$. Note that for some specific parameters, if we set B as before, we have $B - \mathsf{B}_{\mathsf{scale}} < 0$ and so, p_{L-1} as in Equation (11), becomes negative. Thus, we slightly modify B setting

$$B \sim \frac{2\tau\omega\sqrt{L-1}\log_\omega(\mathsf{B}_{\mathsf{ks}})\mathsf{B}_{\mathsf{ks}}}{\mathsf{B}_{\mathsf{const}}} + \mathsf{B}_{\mathsf{scale}} \tag{14}$$

Previous work has shown, that $3 \leq \omega \leq 5$ is a good choice for ω [13].

No rotations. If $\tau = 0$, Equation (12) becomes

$$\frac{\xi B^2}{p_\ell} + \frac{\omega\mathsf{B}_{\mathsf{ks}}\sqrt{\lfloor \log_\omega(q_\ell)\rfloor}}{p_\ell} + \mathsf{B}_{\mathsf{scale}} < B.$$

As before, we must have a positive discriminant and thus

$$1 - \frac{4\xi}{p_\ell}\Big(\frac{\omega \mathsf{B}_{\mathsf{ks}}\sqrt{(\ell+1)\log_\omega(p_\ell)}}{p_\ell} + \mathsf{B}_{\mathsf{scale}}\Big) \sim 1 - \frac{4\xi}{p_\ell}\mathsf{B}_{\mathsf{scale}} \geq 0.$$

We set $p_1 \sim \ldots \sim p_{L-2} \sim 4\xi\mathsf{B}_{\mathsf{scale}}$ and $B \sim \frac{1}{2\xi/p_\ell} \sim 2\mathsf{B}_{\mathsf{scale}}$.

BV-RNS key-switching. Applying the RNS variant of the BV key switching, the situation is more complex. Indeed, after the modulus switching procedure, we have

$$\frac{(\varepsilon B + \eta\tau p_\ell\sqrt{\ell+1}\mathsf{B}_{\mathsf{ks}})^2}{p_\ell} + \frac{\mathsf{B}_{\mathsf{ks}}p_\ell\sqrt{\ell+1}}{p_\ell} + \mathsf{B}_{\mathsf{scale}} < B \tag{15}$$

which (in the variable B) must have a positive discriminant: $1 - 4\varepsilon\eta\tau\sqrt{\ell+1}\mathsf{B}_{\mathsf{ks}} - (4\varepsilon^2\mathsf{B}_{\mathsf{ks}}\sqrt{\ell+1} + 4\varepsilon^2\mathsf{B}_{\mathsf{scale}})/p_\ell \geq 0$. But this is impossible. This means that the error has grown too much, so we have to reduce it again by applying (another) modulus switching. More precisely, from the level ℓ, we directly go down to level $\ell - 2$, and since the p_ℓ's have roughly the same size, we have

$$\frac{(\varepsilon B + \eta\tau p_\ell\sqrt{\ell+1}\mathsf{B}_{\mathsf{ks}})^2}{p_\ell^2} + \frac{\mathsf{B}_{\mathsf{ks}}p_\ell\sqrt{\ell+1}}{p_\ell^2} + \mathsf{B}_{\mathsf{scale}} < B.$$

As before we must have a positive discriminant, namely,

$$1 - \frac{4\varepsilon\eta\tau\sqrt{\ell+1}\mathsf{B}_{\mathsf{ks}}}{p_\ell} - \frac{4\varepsilon^2\mathsf{B}_{\mathsf{ks}}\sqrt{\ell+1}}{p_\ell^3} - \frac{4\varepsilon^2\mathsf{B}_{\mathsf{scale}}}{p_\ell^2} \geq 0,$$

thus we have $p_\ell \sim 4\varepsilon\eta\tau\sqrt{\ell+1}\mathsf{B}_{\mathsf{ks}}$. Since for each multiplication we need to use 2 moduli, we have to double the number of moduli p_ℓ. So we have $p_1 \sim \ldots \sim p_{L-2} \sim 4\varepsilon\eta\tau\sqrt{L-1}\mathsf{B}_{\mathsf{ks}}$, where $L = 2(M+1)$ with M the multiplicative depth of the circuit[4]. Since $2\varepsilon\eta\tau\sqrt{\ell+1}\mathsf{B}_{\mathsf{ks}}/p_\ell \leq 1/2$ and the discriminant is between 0 and 1, we have

$$B \sim \frac{1}{\xi/p_\ell^2} \sim 16\eta^2\tau^2(L-1)\mathsf{B}_{\mathsf{ks}}^2. \tag{16}$$

No rotations. If $\tau = 0$, Equation (15) becomes

$$\frac{\xi B^2}{p_\ell} + \frac{\mathsf{B}_{\mathsf{ks}}p_\ell\sqrt{\ell+1}}{p_\ell} + \mathsf{B}_{\mathsf{scale}} < B.$$

As before, imposing a positive discriminant, we obtain, for any $1 \leq \ell \leq L-2$, $p_\ell \sim 4\xi(\sqrt{L-1}\mathsf{B}_{\mathsf{ks}} + \mathsf{B}_{\mathsf{scale}})$ and $B \sim \frac{1}{2\xi/p_\ell} \sim 2(\sqrt{L-1}\mathsf{B}_{\mathsf{ks}} + \mathsf{B}_{\mathsf{scale}})$.

[4] We are going to see that also for p_0 we need the error reduction.

GHS key-switching. After the first step of the key switching, the error grows from $(\varepsilon B+\eta\tau v_{ks})^2$ to either $P(\varepsilon B+\eta\tau v_{ks})^2+q_\ell \mathsf{B_{ks}}$ (if we use GHS key-switching) or $P(\varepsilon B+\eta\tau v_{ks})^2+\sqrt{\ell+1}q_\ell \mathsf{B_{ks}}$ (when we execute its RNS variant). Then, we perform the modulus switching directly down to a smaller modulus decreasing the noise by $1/(Pp_\ell)$ and adding the error $\mathsf{B_{scale}}$ (or $\sqrt{k}\mathsf{B_{scale}}$ in the RNS case). As before, we want to reduce the noise size back to B, so we set

$$\frac{P(\varepsilon B+\eta\tau v_{ks})^2+f_0^{ks}\mathsf{B_{ks}}}{Pp_\ell}+f_1^{ks}\mathsf{B_{scale}}<B,$$

where $\mathsf{B_{scale}}$, v_{ks} and f_i^{ks} as in Equations (7), (9) and (10), respectively. For $\xi=\varepsilon^2$, the equation above becomes

$$\begin{aligned}&\tfrac{\xi B^2}{p_\ell}+\tfrac{2\eta\tau\varepsilon}{p_\ell}\Big(\tfrac{f_0^{ks}}{P}\mathsf{B_{ks}}+f_1^{ks}\mathsf{B_{scale}}\Big)B+\tfrac{\eta^2\tau^2}{p_\ell}\Big(\tfrac{f_0^{ks}}{P}\mathsf{B_{ks}}+f_1^{ks}\mathsf{B_{scale}}\Big)^2+\\&\tfrac{f_0^{ks}}{Pp_\ell}\mathsf{B_{ks}}+f_1^{ks}\mathsf{B_{scale}}<B.\end{aligned}\tag{17}$$

To solve this inequality in B, we follow the idea of Gentry *et al.* [11]. Let

$$R_\ell=\frac{\eta^2\tau^2}{p_\ell}(\frac{f_0^{ks}}{P}\mathsf{B_{ks}}+f_1^{ks}\mathsf{B_{scale}})^2+\frac{f_0^{ks}}{Pp_\ell}\mathsf{B_{ks}}+f_1^{ks}\mathsf{B_{scale}}.$$

Since R_ℓ increases with larger ℓ's, we have to satisfy this inequality for the largest modulus $\ell=L-2$. Moreover, $R_{L-2}>f_1^{ks}\mathsf{B_{scale}}$. Since we want that this term is as close to $\mathsf{B_{scale}}$ as possible, we have to set P large enough. Namely, either

$$P>Kf_0^{ks}\mathsf{B_{ks}}/\mathsf{B_{scale}},\tag{18}$$

for a *large enough* constant $K\in\mathbb{N}$ (i.e., we can take $K\sim 100$). Equation (17) becomes $\frac{\xi B^2}{p_\ell}+(\frac{2\eta\tau\varepsilon f_1^{ks}}{p_\ell}\mathsf{B_{scale}}-1)B+\frac{(\eta\tau f_1^{ks})^2}{p_\ell}\mathsf{B_{scale}^2}+f_1^{ks}\mathsf{B_{scale}}<0$. Thus, to satisfy this equation, we again must have a positive discriminant and therefore

$$\Big(\frac{2\eta\tau\varepsilon f_1^{ks}}{p_\ell}\mathsf{B_{scale}}-1\Big)^2-4\frac{\xi}{p_\ell}\Big(\frac{(\eta\tau f_1^{ks})^2}{p_\ell}\mathsf{B_{scale}^2}+f_1^{ks}\mathsf{B_{scale}}\Big)\geq 0,$$

that is, $1-4f_1^{ks}(\eta^2\tau\mathsf{B_{const}}+\xi)\mathsf{B_{scale}}/p_\ell\geq 0$. We then have

$$p_1\sim\ldots\sim p_{L-2}\sim 4f_1^{ks}(\eta^2\tau\mathsf{B_{const}}+\xi)\mathsf{B_{scale}}\tag{19}$$

and $R_{L-2}\sim f_1^{ks}\mathsf{B_{scale}}$. Finally, if we set p_ℓ as in (19), we have the discriminate equal to zero and we can find B with

$$B\sim f_1^{ks}\Big(\tau/\mathsf{B_{const}}+2\Big)\mathsf{B_{scale}}.\tag{20}$$

No rotations. If $\tau=0$, Equation (17) becomes

$$\frac{\xi B^2}{p_\ell}+\frac{f_0^{ks}\mathsf{B_{ks}}}{Pp_\ell}+f_1^{ks}\mathsf{B_{scale}}<B,$$

where ξ is as in Table 1. Equations (19) and (20) are the same, instead P can be decrease to either $\frac{Kq_{L-3}}{\mathsf{B_{scale}}}\mathsf{B_{ks}}$ in the GHS case or $\frac{Kq_{L-3}\sqrt{L-2}}{\mathsf{B_{scale}}}\mathsf{B_{ks}}$ for GHS-RNS. Indeed, R_ℓ changes to either $\frac{q_{\ell-1}}{P}\mathsf{B_{ks}}+f_1^{ks}\mathsf{B_{scale}}$ (GHS) or $\frac{q_{\ell-1}\sqrt{L-2}}{P}\mathsf{B_{ks}}+f_1^{ks}\mathsf{B_{scale}}$ (GHS-RNS) (see also [11] for the specific case when $\tau=0$ and $\mathsf{B_{clean}}=1$).

Hybrid key switching. The noise after the key switching and the modulus switching is at most

$$\frac{\left(\varepsilon B + \eta\tau(\frac{f_0^{ks}}{P}\mathsf{B_{ks}} + f_1^{ks}\mathsf{B_{scale}})\right)^2}{p_\ell} + \frac{f_0^{ks}}{Pp_\ell}\mathsf{B_{ks}} + f_1^{ks}\mathsf{B_{scale}} < B.$$

Following the same argument as before, we obtain the same equality for P, p_i and B (i.e., as in Equations (18) to (20), respectively). Specifically, for P we have either

$$P \geq K\omega\sqrt{\log_\omega(q_{L-2})}\mathsf{B_{ks}}/\mathsf{B_{scale}},$$

when we use the hybrid key-switching, or, in the RNS case,

$$P \geq Kp_{L-2}^{L/\omega}\sqrt{w(L-1)}\mathsf{B_{ks}}/\mathsf{B_{scale}},$$

for a *large enough* constant $K \in \mathbb{N}$ (i.e., we can take $K \sim 100$).

No rotations. In this case, if we do not have any rotation or any constant multiplication, the equations for P, p_ℓ and B are the same (setting $\tau = 0$ and $\mathsf{B_{const}} = 1$, respectively).

3.7 The Bottom Modulus

The bottom modulus is the last modulus in the prime chain, that is p_0. At level zero, we do not need any modulus reduction. To ensure a correct decryption, we require that the noise bounded by $(\varepsilon B + \eta\tau v_{ks})^2$ is smaller than $q_0/2$ and thus $c_m(\varepsilon B + \eta\tau v_{ks})^2 < q_0/2$, namely,

$$p_0 = q_0 > 2c_m(\varepsilon B + \eta\tau v_{ks})^2. \tag{21}$$

BV key switching. Equation (21) becomes $z > \varepsilon B + \eta\tau\omega\mathsf{B_{ks}}\sqrt{\log_\omega(2c_m z^2)}$ 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

$$z > \varepsilon B + \eta\tau\omega\mathsf{B_{ks}}\left(\sqrt{\log_\omega(2c_m)} + \sqrt{2\log_\omega z}\right) \tag{22}$$

We claim that this inequality holds for $z = 2\varepsilon B$. Indeed, since B is as in Equation (14), then Equation (22) becomes

$$\frac{\varepsilon B}{\eta\tau\omega\mathsf{B_{ks}}} - \sqrt{\log_\omega(2c_m)} \sim 2\sqrt{L-1}\log_\omega(\mathsf{B_{ks}}) + \frac{\mathsf{B_{const}}\mathsf{B_{scale}}}{2\tau\omega\mathsf{B_{ks}}} > \sqrt{2\log_\omega(2\varepsilon B)}.$$

Note that $2\varepsilon B < \mathsf{B_{ks}^3}$, so the previous inequality holds since

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

Namely,

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

If $\tau = 0$, Equation (21) becomes $p_0 > 2c_m\xi B^2 = 8c_m\xi\mathsf{B_{scale}^2}$.

BV-RNS key switching. As before, applying the RNS version of BV key switching, the situation is more complex. Indeed, Equation (21) becomes $p_0 > 2c_m(\varepsilon B+\eta\tau p_0\mathsf{B_{ks}})^2$, which is impossible. This means that the error has grown too much, so we have to reduce it again by applying (another) modulus switching. More precisely, at level zero, we cannot perform any operation. To ensure a correct decryption, we require that the noise is smaller than $q_0/2$ and thus

$$p_0 > 2c_m B = 32c_m\eta^2\tau^2(L-1)\mathsf{B_{ks}^2},$$

since B is as Equation (16).

Different the case when we do not have any rotation. Indeed, Equation (21) becomes $p_0 > 2c_m\xi B^2 = 8c_m\xi(\sqrt{L-1}\mathsf{B_{ks}} + \mathsf{B_{scale}})^2$.

Other cases. Note that $\eta\tau v_{ks} = \eta\tau(f_0^{ks}\mathsf{B_{ks}}/P + f_1^{ks}\mathsf{B_{scale}}) \sim \eta\tau f_1^{ks}\mathsf{B_{scale}}$ and thanks to Equation (20), $\varepsilon B = f_1^{ks}(\eta\tau + 2\eta\mathsf{B_{const}})\mathsf{B_{scale}}$. So

$$p_0 > 2c_m\big(\varepsilon B + \eta\tau v_{ks}\big)^2 \sim 2c_m\big(2\eta f_1^{ks}\mathsf{B_{scale}}(\tau + \mathsf{B_{const}})\big)^2.$$

If we do not have any rotation, Equation (21) becomes $p_0 > 2c_m\xi B^2$, namely, either $p_0 > 8c_m\xi\mathsf{B_{scale}^2}$ in the GHS and Hybrid case, or $p_0 > 8c_m k\xi\mathsf{B_{scale}^2}$ in their RNS variant.

3.8 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 2.

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

Table 2. The key switching modulus P.

From Table 2, 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 $K \sim 100$), $\mathsf{B}_{\mathsf{const}} = Dt\sqrt{n/12}$, $\mathsf{B}_{\mathsf{clean}}$, $\mathsf{B}_{\mathsf{scale}}$ and B_{ks} as in Equations (6) to (8), respectively.

3.9 Parameters Specification

In Table 3 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 3. Required parameter specification

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

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

3.10 Modulus Size for Power-of-Two Cyclotomics

In this section we summarize the moduli size considering $\Phi_m(x) = x^n + 1$ and $n = 2^\kappa$ (and so $m = 2n$). Let ξ be as in Table 1, $\mathsf{B}_{\mathsf{const}} = Dt\sqrt{n/12}$, $\mathsf{B}_{\mathsf{clean}}$, $\mathsf{B}_{\mathsf{scale}}$ and B_{ks} as in Equations (6) to (8), respectively, with $\phi(m) = n$. ks denote the key switching procedure, where B, G/H are BV, GHS and Hybrid without RNS variant, respecively.

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

Table 4. Case with no rotations ($\tau = 0$).

Note that, in the RNS variant of BV key-switching when $\tau \neq 0$ (denoted by B*) $L = 2(M+1)$, where M is the multiplicative depth of the circuit. In all the other cases $L = M + 1$.

p_0	$8\eta^2(2\tau\omega\sqrt{L-1}\log_\omega(\mathsf{B_{ks}})\mathsf{B_{ks}} + \mathsf{B_{const}}\mathsf{B_{scale}})^2$	B
	$32\eta^2\tau^2(L-1)\mathsf{B_{ks}^2}$	B*
	$8\eta^2(\tau + \mathsf{B_{const}})^2\mathsf{B_{scale}^2}$	G/H
	$8k\eta^2(\tau + \mathsf{B_{const}})^2\mathsf{B_{scale}^2}$	G/H-RNS
p_ℓ	$\eta^2\mathsf{B_{const}}(4\tau\omega\sqrt{L-1}\log_\omega(\mathsf{B_{ks}})\mathsf{B_{ks}} + \mathsf{B_{const}}\mathsf{B_{scale}})$	B
	$4\eta^2\tau\sqrt{L-1}\mathsf{B_{const}}\mathsf{B_{ks}}$	B*
	$4\eta^2(\tau + \mathsf{B_{const}})\mathsf{B_{const}}\mathsf{B_{scale}}$	G/H
	$4\eta^2\sqrt{k}(\tau + \mathsf{B_{const}})\mathsf{B_{const}}\mathsf{B_{scale}}$	G/H-RNS
p_{L-1}	$\left(2\tau\omega\sqrt{L-1}\log_\omega(\mathsf{B_{ks}})\mathsf{B_{ks}}\right)^{-1}\mathsf{B_{const}}\mathsf{B_{clean}}$	B
	$\left(16\eta^2\tau^2(L-1)\mathsf{B_{ks}^2} - \mathsf{B_{scale}}\right)^{-1}\mathsf{B_{clean}}$	B*
	$\left(\frac{\tau}{\mathsf{B_{const}}} + 1\right)^{-1}\frac{\mathsf{B_{clean}}}{\mathsf{B_{scale}}}$	G/H
	$\left(\sqrt{k}\left(\frac{\tau}{\mathsf{B_{const}}} + 2\right) - 1\right)^{-1}\frac{\mathsf{B_{clean}}}{\mathsf{B_{scale}}}$	G/H-RNS

Table 5. Case with $\tau \neq 0$ rotations.

4 A Parameter Generator for BGV

In the following, we introduce the empirically derived, closed security formula for our parameter generator. Then, we introduce the generator itself within the context of our research.

4.1 Security Analysis

We propose an empirically derived formula linking the security level λ with the dimension n for a given ciphertext modulus size $\log q$ enabling a fast security

estimate for the 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 semples M, namely the subdimension, such that $||\mathbf{b}_1|| = \delta_0^M q^{n/M}$ is minimized. Micciancio and Regev [17] 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

$$n = \log q / (4 \log \delta_0). \tag{23}$$

Substituting (23) in (5), we have a bound linking λ, n and q. Then, we run through the following steps deriving our final formula

1. We run the lattice estimator [3] for the dimensions $n = 2^k$ and 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 [2].
2. Starting from the theoretical bound linking λ, n, and q, we find a function that follows the data points generated with the lattice estimator.
3. Finally, we model the resulting formula with coupled optimization finding the final constants.

Thus, we obtain

$$\lambda \approx -\log\left(\frac{\mathrm{A}\log q}{n}\right)\frac{\mathrm{B}n}{\log q} + \mathrm{C}\sqrt{\frac{\log q}{n}}\log\left(\frac{n}{\log q}\right), \tag{24}$$

where

$\mathrm{A} = 0.07$ $\mathrm{B} = 0.34$ $\mathrm{C} = 18.53$ when $\chi_s = \mathcal{U}_3$ (Figure 2).
$\mathrm{A} = 0.65$ $\mathrm{B} = 0.53$ $\mathrm{C} = 22.88$ when $\chi_s = \chi_e$.

This formula (provide λ, receive n) opens up the possibility to generate parameters for a given security level.

We provide a visualization of the formula in Figure 2 together with the original data points by the Lattice Estimator (for $\chi_s = \mathcal{U}_3$). Note that for GHS and Hybrid key switching variants, we have to consider the bigger modulus $Q = qP$ to evaluate the security.

4.2 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 as well as a simple benchmarking setup to easily compare different parameter scenarios on their local setups. The generator itself is written in Python and will be publicly available on GitHub [5].

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

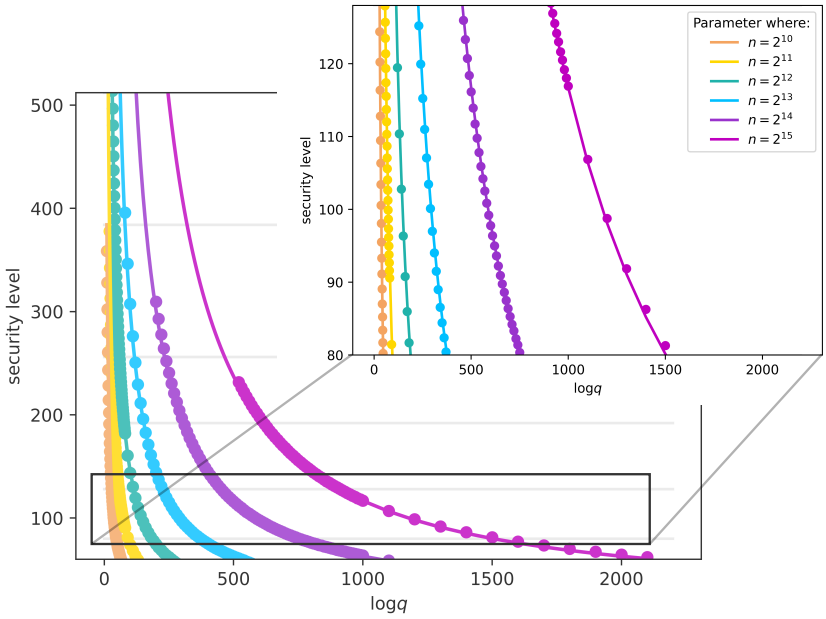

Fig. 2. The security formula with data points of the Lattice Estimator when $\chi_s = \mathcal{U}_3$.

It consists of four modules: the BGV module, a code generation module, a configuration module and an interactive module. The BGV module is the heart of the generator. It provides a function to calculate all necessary bound constants as well as generation functions for the ciphertext and key switching moduli q and P as well as a power-of-two cyclotomic order m. Additionally, it handles the logic of the interactive menu and implements compatibility checks for our target library PALISADE.

The code generation module outputs three files: a Makefile, a `main.cpp` file with example and comparison code for a single plaintext and a `bench.cpp` file with simple benchmarks for the basic homomorphic operations. As we cannot cover all possible environments, users can adjust some of the environment-specific options such as the benchmark repetitions or the include paths for each library in the configuration module. The interactive module handles user dialogs as well as input parsing.

Interactive Mode The interactive mode of the parameter generator prompts the user for a number of questions. We list required inputs in the first part, optional inputs in the second part of Table 6. After providing all required information, the user receives the output in text form and, if the PALISADE 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.

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'`, `'PALISADE'`
Full Batching	full batching with t, `'True'` or `'False'`
Secret Distribution	`'Ternary'`, `'Error'`
Key Switching	`'Hybrid'`, `'BV'`, `'GHS'`
ω	any integer ≥ 1

Table 6. Required and optional inputs to the parameter generator

An exemplary output is the following.

```
$ python fhegen/bgv.py
Welcome to the interactive parameter generator for BGV! :)

Do you want a specific plaintext modulus? [N/?]:
[...]
Do you want to continue to the advanced settings? [N/y]:

Generated your BGV configuration!
sec:   157.49
d:     16384
t:     65537
logq:  354 (111, 80, 3)
logP:  11
slots: 16384
Generating Makefile, main.cpp and bench.cpp for PALISADE.
```

Listing 1.2. Shortened example on usage and output of the parameter generator

Limitations Due to internal workings of PALISADE, we cannot guarantee that all parameter sets work. For example, PALISADE supports only a plaintext of up to 60 bit and 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.

Future Work For future work, we would propose extending the parameter generator to other available BGV implementations such as the recently released OpenFHE[6]. Additionally, we suggest applying our approach of flexibel parameter generation to other FHE schemes such as Brakerski Fan Vercauteren (BFV). In general, we think that there still is room to improve the parameter generation process for FHE schemes in use-case specific scenarios.

[6] `https://www.openfhe.org/`

5 Conclusion

Finding an optimal set of parameters for a specific FHE scheme is a challenging task. Use-case specific aspects such as the multiplicative depth or the amount of rotations with key switchings performed significantly impact the error growth. Hence, evaluating parameters in each use case is a necessity for choosing FHE parameters.

In this work, we extend previous analysis bringing together the DCRT representation and the canonical embedding norm and improve upon the existing state-of-the-art. Additionally, we define new circuit models that include essential BGV operations such as constant multiplication or rotations. We then provide an in-depth analysis of the noise growth for the three main variants of key switching and provide several formulas for parameter generation.

We also provide an empirically derived and closed formula to estimate the security of a given parameter set based using coupled optimization for ternary and Gaussian secrets. 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 PALISADE.

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

Bibliography

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

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

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

[4] Brakerski, Z., Vaikuntanathan, V.: Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. pp. 505–524. Springer, Berlin, Heidelberg (2011)

[5] Chen, H., Kim, M., Razenshteyn, I., Rotaru, D., Song, Y., Wagh, S.: Maliciously secure matrix multiplication with applications to private deep learning. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 31–59. Springer (2020)

[6] Costache, A., Smart, N.P.: Which ring based somewhat homomorphic encryption scheme is best? In: Cryptographers' Track at the RSA Conference. pp. 325–340. Springer (2016)

[7] Costache, A., Laine, K., Player, R.: Evaluating the effectiveness of heuristic worst-case noise analysis in FHE. In: European Symposium on Research in Computer Security. pp. 546–565. Springer (2020)

[8] Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Annual Cryptology Conference. pp. 643–662. Springer (2012)

[9] Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008. pp. 31–51. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

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

[11] Gentry, C., Halevi, S., Smart, N.P.: Homomorphic Evaluation of the AES Circuit. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. pp. 850–867. Springer, Berlin, Heidelberg (2012)

[12] Iliashenko, I.: Optimisations of fully homomorphic encryption (2019)

[13] Kim, A., Polyakov, Y., Zucca, V.: Revisiting homomorphic encryption schemes for finite fields (2021)

[14] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)

[15] Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., Fitzek, F.H., Aaraj, N.: Survey on fully homomorphic encryption, theory and applications (2022)
[16] Martins, P., Sousa, L., Mariano, A.: A survey on fully homomorphic encryption: An engineering perspective. ACM Computing Surveys (CSUR) **50**(6), 1–33 (2017)
[17] Micciancio, D., Regev, O.: Lattice-based Cryptography. Springer, Berlin, Heidelberg (2009)
[18] PALISADE: `https://palisade-crypto.org`
[19] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing. pp. 84–93 (2005)
[20] Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical programming **66**(1-3), 181–199 (1994)
[21] Seiler, G.: Faster avx2 optimized ntt multiplication for ring-lwe lattice cryptography (January 2018), `https://eprint.iacr.org/2018/039`