Leveled Homomorphic Encryption Schemes for Homomorphic Encryption Standard

Shuhong Gao[1] and Kyle Yates[1]

[1]School of Mathematical and Statistical Sciences, Clemson University
sgao@clemson.edu, kjyates@clemson.edu

Abstract

Homomorphic encryption allows for computations on encrypted data without exposing the underlying plaintext, enabling secure and private data processing in various applications such as cloud computing and machine learning. This paper presents a comprehensive mathematical foundation for three prominent homomorphic encryption schemes: Brakerski-Gentry-Vaikuntanathan (BGV), Brakerski-Fan-Vercauteren (BFV), and Cheon-Kim-Kim-Song (CKKS), all based on the Ring Learning with Errors (RLWE) problem. We align our discussion with the functionalities proposed in the recent homomorphic encryption standard, providing detailed algorithms and correctness proofs for each scheme. Additionally, we propose improvements to the current schemes focusing on noise management and optimization of public key encryption and leveled homomorphic computation. Our modifications ensure that the noise bound remains within a fixed function for all levels of computation, guaranteeing correct decryption and maintaining efficiency comparable to existing methods. The proposed enhancements reduce ciphertext expansion and storage requirements, making these schemes more practical for real-world applications.

Keywords. Homomorphic Encryption, Learning with Errors, Ring Learning with Errors, Noise Bounds, Lattice Attacks.

1 Introduction

Homomorphic encryption describes encryption schemes that allow for addition and multiplication operations to be performed on ciphertexts without needing or leaking any information about the secret key or user messages. Furthermore, the operations in the ciphertext space correspond to performing the same operations on the original messages, which can be performed by any third party with knowledge of only the public information. Homomorphic encryption has several modern applications, such as secure cloud computing and private machine learning. With Craig Gentry's work in 2009 [1], provably secure homomorphic encryption became viable using ideal lattices. This construction closely relates to the commonly used learning with errors (LWE) problem, with the hardness of LWE being a result proved by Regev in 2005 [2].

Three of the most common modern homomorphic encryption schemes are based on a ring version of LWE problems, known as the ring learning with errors (RLWE) problems [3]. These schemes are the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [4, 5], the Brakerski-Fan-Vercauteren (BFV) scheme [6], and the Cheon-Kim-Kim-Song (CKKS) scheme [7]. BGV and BFV both allow for homomorphic computation for exact arithmetic, while CKKS provides homomorphic computation for numerical computation with certain accuracy. With recent efforts to standardize homomorphic encryption schemes and security [8, 9], it is desirable to have concrete and mathematically solid discussions on encryption schemes and homomorphic computing protocols that match the functionalities proposed in the standard, including parameter specifications for efficiency and security.

We should mention that Chillotti et al. [10, 11] present a fully homomorphic encryption scheme that can perform one bit operation in less than 0.1 second, while Gao [12] and Case et al. [13] present a fully homomorphic encryption scheme with similar running time but a much smaller ciphertext expansion (< 20). However, each operation in these schemes is prohibitively expensive at the moment. Leveled schemes have a much larger ciphertext expansion, but each operation is much cheaper. A leveled scheme allows for some predetermined number of operations [5, 14, 15], which is the style we opt for in this paper. Several works study noise bounds for homomorphic encryption [14, 16, 17, 18, 19, 20] in both the canonical embedding and infinity norms. These analyses include both theory and implementations. Speedups can be implemented via the residue number system (RNS) [21, 22], which uses the Chinese remainder theorem to break down computations into smaller rings. The schemes we present in this paper can all be implemented using RNS representation.

Our Contributions. This paper has two main goals. The first goal is to present detailed algorithms for the functionalities proposed in the homomorphic encryption standard [8, 9] for each of the BFV, BGV, and CKKS schemes, and present a detailed correctness proof for all the functionalities. This lays a rigorous mathematical foundation for homomorphic encryption schemes. The second goal is to improve the current schemes for BFV, BGV, and CKKS. We present modified schemes for each of the three schemes, especially in public key encryption and leveled homomorphic computing, and focus on noise control and the worst-case noise bounds, thereby reducing ciphertext expansion and storage expenses. In particular, under the modified schemes, the noise bound for ciphertexts from public key encryption and from homomorphic computing at each level is always bounded by a fixed function ρ, which depends on the underlying ring. The worst-case bound guarantees that ciphertexts can always be decrypted correctly, with no probability of decryption error, which is preferred for many applications. Furthermore, parameter sizes resulting from our worst-case bounds are comparable to parameter sizes from average-case bounds in the literature, thus not degrading the efficiency of the schemes.

Organization of This Paper. In Section 2 we describe notations and necessary background. We then introduce LWE and RLWE problems. We present and prove

two variations of modulus reduction, which is later applied to RLWE-based encryption schemes. In Section 3 we outline three RLWE-based homomorphic encryption schemes: BFV, BGV, and CKKS. For these three schemes, we provide modified encryption to better control noise and conduct a thorough worst case theoretical noise analysis. In Section 4 we discuss leveled schemes and present techniques in choosing parameters to guarantee homomorphic operations. We also outline operations in RNS here. In Section 5 we give a brief discussion of attack techniques for LWE problems. In Section 6 we provide concluding remarks and further potential research topics. Appendix A contains the proofs of the lemmas on correctness of functionalities for all the algorithms.

2 Notations and Preliminaries

2.1 Notations

For a positive integer q, define $\mathbb{Z}_q := \mathbb{Z} \cap (-q/2, q/2]$ to be the ring of centered representatives modulo q. For an element $v \in \mathbb{Z}$, we denote $[v]_q$ to be the modular reduction of v into the interval $\mathbb{Z}_q$ such that q divides $v - [v]_q$. When v is a vector or a polynomial, $[v]_q$ means reducing each entry or coefficient of v modulo q, respectively. Denote R_n as the ring

$$R_n = \mathbb{Z}[x]/(\phi(x))$$

where $\phi(x)$ is a polynomial of degree n and $(\phi(x))$ is the ideal generated by $\phi(x)$. Often, we will choose $\phi(x)$ to be a power of two cyclotomic polynomial. That is, a polynomial of the form $\phi(x) = x^n + 1$, where n is a power of two. For an integer q, we define $R_{n,q}$ as

$$R_{n,q} = \mathbb{Z}_q[x]/(\phi(x)) \cong \mathbb{Z}[x]/(\phi(x), q)$$

where $(\phi(x), q)$ is the ideal generated by $\phi(x)$ and q. When v is a polynomial, $[v]_{\phi(x)}$ denotes modular reduction of the polynomial into R_n. Similarly, when v is a polynomial with integer coefficients, $[v]_{\phi(x),q}$ denotes modular reduction of the polynomial into $R_{n,q}$, where all the coefficients are in $(-q/2, q/2]$.

For a vector or polynomial v, the infinity norm of v, denoted $\|v\|_\infty$, is the maximum entry or coefficient in absolute value of v. Equivalently, if $v = (a_0, \ldots, a_{n-1})$ or $v = \sum_{i=0}^{n-1} a_i x^i$, then

$$\|v\|_\infty = \max\{|a_i| : i = 0, \ldots, n-1\}.$$

$\|\cdot\|_2$ denotes the standard 2-norm. The symbols $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ will denote floor and ceiling respectively, whereas $\lfloor\cdot\rceil$ will denote rounding to the nearest integer, rounding down in the case of a tie. When applying $\lfloor\cdot\rfloor$, $\lceil\cdot\rceil$, or $\lfloor\cdot\rceil$ to a polynomial or vector, we mean the rounding of each entry or coefficient. Define the expansion factor δ_R of R_n as

$$\delta_R = \max\left\{ \frac{\|uv\|_\infty}{\|u\|_\infty \|v\|_\infty} : u, v \in R_n \right\},$$

where uv must be reduced modulo $\phi(x)$ before applying the norm. When $\phi(x) = x^n+1$ where n is a power of two, it is well known that $\delta_R = n$.

2.2 Noise Distributions and Learning With Errors Problems

For a set S, we denote $\chi(S)$ as an arbitrary probability distribution χ on S. We denote U as the uniform distribution. Let $\rho > 0$ be any integer. We denote χ_ρ as any probability distribution on R_n, where each coefficient is random in $[-\rho, \rho]$ and independent. We call χ_ρ an *error distribution* or *noise distribution*. We allow for flexibility in the exact choice of χ_ρ, but most commonly, χ_ρ is chosen as uniform random on $[-\rho, \rho]$, or a truncated discrete Gaussian in order to maintain security [23, 24]. Over $\mathbb{Z}$, the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\alpha q}$ assigns a probability proportional to $\exp(-\pi x^2/(\alpha q)^2)$ for each $x \in \mathbb{Z}$ with standard deviation $\sigma = \alpha q/\sqrt{2\pi}$ [6]. For the cyclotomic polynomial $\phi(x) = x^n + 1$, an n-dimensional extension of $\mathcal{D}_{\mathbb{Z},\alpha q}$ to R_n can be constructed by process of sampling each coefficient from $\mathcal{D}_{\mathbb{Z},\alpha q}$. More details on the impact of the error distribution on security can be found in Section 5.

LWE Problems. For any secret $s \in \mathbb{Z}_q^n$, we sample $e \leftarrow \chi(\mathbb{Z})$ from some desired distribution χ such that $\|e\|_\infty \leq \rho$, where ρ is a desired parameter, we sample a uniform random $a \leftarrow U(\mathbb{Z}_q^n)$, and calculate b via $b := [-\langle a, s\rangle + e]_q$. The ordered pair $(a, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ is called an *LWE sample*. The Search-LWE problem is to find s given many LWE samples. The Decision-LWE problem is given many samples that are either LWE samples or sampled at uniform random from $\mathbb{Z}_q^n \times \mathbb{Z}_q$, decide which distributions the samples are drawn from [25].

When drawing elements from distributions on R_n and $R_{n,q}$, we can similarly define the RLWE problems. For a secret $s \in R_{n,q}$, sample a polynomial $e \leftarrow \chi_\rho$, sample $a \leftarrow U(R_{n,q})$, and compute $b \in R_{n,q}$ via $b := [-as + e]_{\phi(x),q}$. The ordered pair $(a, b) \in R_{n,q}^2$ is called an *RLWE sample*. The RLWE problems can be defined in an analogous way to the LWE problems. Throughout this paper, when given an RLWE sample or similarly structured ordered pair, we will commonly refer to the polynomial e as *the noise term* and $\|e\|_\infty$ as *the noise size*.

Most leveled homomorphic encryption schemes use RLWE as opposed to LWE. The ciphertexts of homomorphic encryption schemes discussed will all essentially take the form of a modified RLWE sample. Regev originally showed the hardness of the LWE problem in [2], which serves as foundation for the security of homomorphic encryption schemes. We discuss more specifics on security in Section 5.

We remark that, in our schemes, we use noise size $\rho = n$, and the noise distribution can be uniform on integers bounded by ρ, or a discrete Gaussian distribution but truncated at ρ. When using a bounded uniform distribution, our choice of noise has standard deviation σ of about $n/\sqrt{3}$. In comparison, most implementations in practice (including the homomorphic encryption standard) use a discrete Gaussian distribution with $\sigma \approx 3.2$ [8, 9, 26, 19]. Our larger noise bound increases the security, which will be discussed later.

2.3 Modulus Reductions

Let Q and q be any positive integers. Given an RLWE sample $(a_0, b_0) \in R_{n,Q}^2$, where $b_0 \equiv -a_0 s + e_0 \mod (\phi(x), Q)$, we can compute a new RLWE sample $(a_0', b_0') \in R_{n,q}^2$ satisfying $b_0' \equiv -a_0' s + e_0' \mod (\phi(x), q)$ for some new noise term e_0'. Although this is a new RLWE sample with a new integer modulus q, the key observation is that the polynomial s remains the same. Furthermore, if given an initial bound on e_0, we can guarantee a bound on e_0' dependent on Q and q. Algorithm 1 gives the procedure for modulus reduction, while Lemma 2.1 shows correctness and the resulting bound on e_0'. Though Q and q are any positive integers, we typically choose $Q > q$ and refer to this procedure as a *modulus reduction* rather than a modulus switch as many other papers do. Note here that we use a subscript of 0 in our RLWE sample, as it will provide more consistency with our later applications of this algorithm to ciphertexts. For this reason, we also label (a_0, b_0) as $\mathtt{ct}_0$ in Algorithm 1.

	$\mathtt{BFV.Modreduce}(\mathtt{ct}_0, Q, q)$
Input:	$Q \in \mathbb{N}$ an integer,
	$q \in \mathbb{N}$ an integer,
	$\mathtt{ct}_0 = (a_0, b_0) \in R_{n,Q}^2$.
Output:	$\mathtt{ct}_0' = (a_0', b_0') \in R_{n,q}^2$.
Step 1.	Compute $a_0' := \lfloor \frac{q a_0}{Q} \rceil$ and $b_0' := \lfloor \frac{q b_0}{Q} \rceil$.
Step 2.	Return $\mathtt{ct}_0' = (a_0', b_0') \in R_{n,q}^2$.

Algorithm 1: BFV Modulus Reduction

Lemma 2.1 *Suppose the input* $\mathtt{ct}_0$ *of Algorithm 1 is an RLWE sample such that* $\|e_0\|_\infty \leq E$. *Let* $\mathtt{ct}_0'$ *be the output of Algorithm 1. Then,*

$$b_0' \equiv -a_0' s + e_0' \mod (\phi(x), q)$$

and $\|e_0'\|_\infty \leq \frac{q}{Q} E + \frac{\delta_R \|s\|_\infty + 1}{2}$. *Furthermore, if* $Q/q > \frac{2E}{\delta_R \|s\|_\infty - 1}$, *then* $\|e_0'\|_\infty < \delta_R \|s\|_\infty$.

What will be more useful than a modulus reduction for a generic RLWE sample will be a modulus reduction for a "modified" RLWE sample that takes the form of a standard BFV ciphertext, hence the algorithm name `BFV.Modreduce`. By a BFV ciphertext, we mean that our input for Algorithm 1 $\mathtt{ct}_0 = (a_0, b_0) \in R_{n,Q}^2$ satisfies

$$b_0 + a_0 s \equiv D_Q m_0 + e_0 \mod (\phi(x), Q)$$

for some noise term $e_0 \in R_n$, where $m_0 \in R_{n,t}$ and D_Q is a positive integer. This format is further clarified in Section 3.1. Classic BFV [6] uses $D_Q = \lfloor Q/t \rfloor$. In this

paper we will always assume $t|(Q-1)$ for any given ciphertext modulus Q, meaning that $D_Q = (Q-1)/t$ when using the same floor definition as in classic BFV. The resulting output $\mathtt{ct}'_0 = (a'_0, b'_0) \in R^2_{n,q}$ of Algorithm 1 satisfies $b'_0 + a'_0 s \equiv D_q m_0 + e'_0 \mod (\phi(x), q)$ for some e'_0, where $D_q = (q-1)/t$ when $t|(q-1)$. Algorithm 1 and the proof of Lemma 2.2 have a similar style to the proof of Lemma 2.3 in [12].

Lemma 2.2 *Suppose the input of Algorithm 1 is a BFV ciphertext such that* $\|e_0\|_\infty \leq E$. *Let* $\mathtt{ct}'_0$ *be the output of Algorithm 1. If* $t|(Q-1)$ *and* $t|(q-1)$, *then*

$$b'_0 + a'_0 s \equiv D_q m_0 + e'_0 \mod (\phi(x), q)$$

and $\left\|e'_0\right\|_\infty \leq \frac{q}{Q}E + 1 + \frac{\delta_R \|s\|_\infty}{2}$. *Furthermore, if* $Q/q > \frac{2E}{\delta_R \|s\|_\infty - 2}$, *then* $\left\|e'_0\right\|_\infty < \delta_R \|s\|_\infty$.

The final reformulation essentially states that if Q/q meets a specific bound depending on E, modulus reduction always produces a new noise term e'_0 bounded by $\delta_R \|s\|_\infty$. A similar algorithm and lemma can also be constructed for a standard BGV ciphertext [4, 5], which is an ordered pair $\mathtt{ct}_0 = (a_0, b_0) \in R^2_{n,Q}$ satisfying

$$b_0 + a_0 s \equiv m_0 + t e_0 \mod (\phi(x), Q)$$

for some noise term e_0 and given message $m_0 \in R_{n,t}$, which is the message space for some integer $t > 1$. The procedure for computing the new ciphertext differs from the previous two modulus reduction algorithms. In particular, we use the procedure outlined in Lemma 4.3.1 of [25], which is given here as Algorithm 2.

	$\mathtt{BGV.Modreduce}(\mathtt{ct}_0, Q, q)$
Input:	$Q \in \mathbb{N}$, an integer,
	$q \in \mathbb{N}$, an integer,
	$\mathtt{ct}_0 = (a_0, b_0) \in R^2_{n,Q}$, BGV ciphertext.
Output:	$\mathtt{ct}'_0 = (a'_0, b'_0) \in R^2_{n,q}$, BGV ciphertext.
Step 1.	Compute
	$\omega_a := [-a_0 q t^{-1}]_Q \quad$ and $\quad \omega_b := [-b_0 q t^{-1}]_Q$.
Step 2.	Compute
	$a'_0 := \left[\frac{q\, a_0 + t\, \omega_a}{Q}\right]_q$ and $b'_0 := \left[\frac{q\, b_0 + t\, \omega_b}{Q}\right]_q$.
Step 3.	Return $\mathtt{ct}'_0 = (a'_0, b'_0) \in R^2_{n,q}$

Algorithm 2: BGV Modulus Reduction

Lemma 2.3 *Suppose the input of Algorithm 2 is a BGV ciphertext such that* $\|e_0\|_\infty \leq E$. *Let* $\mathtt{ct}'_0$ *be the output of Algorithm 2. If* $t|(Q-1)$ *and* $t|(q-1)$, *then*

$$b'_0 + a'_0 s \equiv m_0 + te'_0 \mod (\phi(x), q)$$

and $\left\|e'_0\right\|_\infty \leq \frac{q}{Q}E + 1 + \frac{\delta_R \|s\|_\infty}{2}$. *Furthermore, if* $Q/q > \frac{2E}{\delta_R \|s\|_\infty - 2}$, *then* $\left\|e'_0\right\|_\infty < \delta_R \|s\|_\infty$.

3 Homomorphic Encryption Schemes and Noise Bounds

Most homomorphic encryption schemes in the literatue use a modified version of RLWE to hide messages. In this section, we'll cover three main schemes: BFV [6], BGV [4, 5], and CKKS [7]. For these three schemes, we present modified versions where the noise sizes are improved and always controlled by a fixed bound, namely $\rho = \delta_R \|s\|_\infty$.

Overview of Specifications. Before outlining our specific algorithms, we first provide an overview of specifications for parameters and spaces. Although there are variations between the schemes, the parameter choices outlined below work for all of BFV, BGV, and CKKS (when applicable). These parameter conditions ensure proper functionality regarding homomorphic computation for each scheme. Further caution must be taken when choosing parameters in practice to ensure security, which is discussed in Section 5.

Specifications for Homomorphic Encryption Schemes

`Public Parameters:`	$q \in \mathbb{N}$
	$p_0 \in \mathbb{N}$ with $p_0 \geq 5\delta_R + 3$
	$p_1 \in \mathbb{N}$ with $p_1 \geq 6q$
	$t \in \mathbb{N}$ with $t\|(q-1), t\|(p_0-1),$ and $t\|(p_1-1)$
`Plaintext:`	$m \in R_{n,t}$ for BFV and BGV
	$m \in \mathbb{C}^{n/2}$ for CKKS
`Secret Key:`	$\mathtt{sk} \in R_{n,3}$
`Public Key:`	$\mathtt{pk} \in R^2_{n,p_0 q}$
`Evaluation Key:`	$\mathtt{ek} \in R^2_{n,p_1 q}$
`Ciphertext:`	$\mathtt{ct} \in R^2_{n,q}$
`Noise Bound:`	$\rho = \delta_R \|s\|_\infty$
`Noise Distribution:`	χ_ρ, a probabilistic distribution on R_n with each coefficient random in $[-\rho, \rho]$

We want to emphasize that each coefficient of the distribution χ_ρ can be uniform random on $[-\rho, \rho]$, or any subgaussian truncated by the bound ρ, or any other distribution that is bounded by ρ. All the noise bounds in this paper will be valid, since they depend only on the worst-case bound ρ. We will simply say "Sample $e \leftarrow \chi_\rho$" in all the algorithms for this generic distribution.

The bound ρ appears prominently in our algorithms. It is not just a bound on the noise distribution, but also a worst-case bound for both fresh ciphertexts from public key encryption and new ciphertexts after modulus reduction when going from one level to the next level, as indicated by the lemmas in the previous section. In practical implementations, we often choose n to be a power of 2 and $\phi(x) = x^n + 1$, hence $\delta_R = n$. When $\|s\|_\infty = 1$, we have $\rho = \delta_R \|s\|_\infty = n$.

Remark on message encoding and choice of t. In the BFV scheme, we choose to encode a message polynomial m as $D_q m$ where $D_q = (q-1)/t$, which requires that $q - 1$ is divisible by t. The paper [16] proposes to encode m as $\lfloor qm/t \rceil$, which works for any t and q, hence no restriction that $t|(q-1)$. An extra small amount of noise is introduced from their encryption style, but has minimal impact and gives about the same bounds in our lemmas in the case of t dividing $q - 1$ that we consider. This alternate encryption style is slightly more expensive from a computation standpoint, as additional rounding operations must be performed as opposed to just integer multiplication. We refer the reader to [16] for more details on plaintext modulus choice and SIMD.

3.1 Modified BFV Scheme

BFV Key Generation. The key generation process we use is slightly different from the standard BFV scheme [6] in that the public key and evaluation key are generated in a larger modulus [6, 14, 16, 19], which will be useful for reducing the noise size in ciphertexts. Algorithm 3 gives the key generation for the BFV keys needed, which is the secret key $\mathtt{sk}$, the public key $\mathtt{pk}$, and the evaluation key $\mathtt{ek}$. Here, $\mathtt{sk}$ is kept secret, while $\mathtt{pk}$ and $\mathtt{ek}$ are published. The public key $\mathtt{pk} = (\mathtt{k}_0, \mathtt{k}_1) \in R^2_{n,p_0q}$ satisfies

$$\mathtt{k}_1 + \mathtt{k}_0 s \equiv e \mod (\phi(x), p_0 q)$$

for noise term $e \leftarrow \chi_\rho$. The evaluation key $\mathtt{ek} = (\mathtt{k}'_0, \mathtt{k}'_1) \in R^2_{n,p_1q}$ satisfies

$$\mathtt{k}'_0 + \mathtt{k}'_1 s \equiv p_1 s^2 + e'_1 \mod (\phi(x), p_1 q)$$

for noise term $e'_1 \leftarrow \chi_\rho$. We remark that in Algorithm 3 we choose s randomly rather than specifying the sampling distribution. This is intentional, as s may be desired to be chosen to satisfy certain properties in practice. For instance, s is often chosen randomly with a predetermined Hamming weight in practice.

BFV Encryption and Decryption. We encrypt a message $m_0 \in R_{n,t}$ using a modified version of the standard public key procedure for BFV. Note that we choose the plaintext modulus t so that t divides $p_0 - 1$ and $q - 1$, and therefore divides

	$\texttt{BFV.Keygen}(q, p_0, p_1)$
Input:	$q \in \mathbb{N}$,
	$p_0 \in \mathbb{N}$ with $p_0 \geq 5\delta_R + 3$,
	$p_1 \in \mathbb{N}$ with $p_1 \geq 6q$.
Output:	$\texttt{sk} = s \in R_{n,3}$ secret key,
	$\texttt{pk} = (\texttt{k}_0, \texttt{k}_1) \in R^2_{n,p_0q}$ public key,
	$\texttt{ek} = (\texttt{k}'_0, \texttt{k}'_1) \in R^2_{n,p_1q}$ evaluation key.
Step 1.	Choose randomly $s \in R_{n,3}$.
Step 2.	Sample $\texttt{k}_0 \leftarrow U(R_{n,p_0q})$ and $e \leftarrow \chi_\rho$.
	Compute $\texttt{k}_1 := [-(\texttt{k}_0 s + e)]_{\phi(x),p_0q}$.
Step 3.	Sample $\texttt{k}'_1 \leftarrow U(R_{n,p_1q})$ and $e'_1 \leftarrow \chi_\rho$.
	Compute $\texttt{k}'_0 := [-\texttt{k}'_1 s + p_1 s^2 + e'_1]_{\phi(x),p_1q}$.
Step 4.	Return $\texttt{sk} = s$, $\texttt{pk} = (\texttt{k}_0, \texttt{k}_1)$, and $\texttt{ek} = (\texttt{k}'_0, \texttt{k}'_1)$.

Algorithm 3: BFV Key Generation

$p_0q - 1$. We immediately reduce the ciphertext modulus from p_0q to q before adding the message bits, then return the ciphertext. The purpose of this is to ensure the noise term on the returned ciphertext $\texttt{ct}'_0$ is within the constant bound of ρ. Given our description of the secret key selection, it is obvious that $\|s\|_\infty = 1$. Most results we provide can be easily modified for the case of general $\|s\|_\infty$ however, allowing for some flexibility in key generation if desired. The exact choices of p_0 and q will of course depend on several factors, such as the desired number of homomorphic computations. We will further discuss the choices of these in Section 4. Assuming these parameters, Algorithm 4 describes the encryption procedure for BFV. In many algorithms, we'll refer to inputs as "BFV ciphertexts". By this, we mean some ordered pair $\texttt{ct} = (a, b) \in R^2_{n,q}$ satisfying

$$b + as \equiv D_q m + e \mod (\phi(x), q)$$

for some message $m \in R_{n,t}$, some noise term $e \in R_n$, ciphertext modulus q, and constant $D_q = \lfloor q/t \rfloor = (q-1)/t$. This encryption style is the classic form of BFV encryption [6]. If $\|e\|_\infty \leq E$, we will say $\texttt{ct} = (a, b)$ is a BFV ciphertext with noise bounded by E.

Lemma 3.1 provides correctness and the corresponding noise bound resulting from encryption. The bounds in Lemma 3.1 are assuming that $\|s\|_\infty = \|u\|_\infty = 1$. However, the bounds can be discussed in terms of more generic u and s, in which case the bound condition on p_0 is $p_0 > \frac{2\delta_R^2(\|u\|_\infty + \|s\|_\infty) + 2\delta_R}{\delta_R \|s\|_\infty - 1}$. In this case, the resulting noise term from encryption still satisfies $\left\|e'_0\right\|_\infty < \rho$.

	$\texttt{BFV.Encrypt}(m_0, D_q, \texttt{pk})$
Input:	$m_0 \in R_{n,t}$ message,
	$D_q \in \mathbb{N}$ constant,
	$\texttt{pk} = (\texttt{k}_0, \texttt{k}_1) \in R^2_{n,p_0q}$ public key.
Output:	$\texttt{ct}'_0 = (a'_0, b'_0) \in R^2_{n,q}$ BFV ciphertext.
Step 1.	Sample $u \leftarrow U(R_{n,3})$ and sample $e_1, e_2 \leftarrow \chi_\rho$.
Step 2.	Compute $(a_0, b_0) \in R^2_{n,p_0q}$ where
	$a_0 := [\texttt{k}_0 u + e_1]_{\phi(x),p_0q}$,
	$b_0 := [\texttt{k}_1 u + e_2]_{\phi(x),p_0q}$.
Step 3.	Compute
	$(a'_0, b^*_0) := \texttt{BFV.Modreduce}((a_0, b_0), p_0q, q)$,
	$b'_0 := [b^*_0 + D_q m_0]_q$.
Step 4.	Return $\texttt{ct}'_0 = (a'_0, b'_0) \in R^2_{n,q}$.

Algorithm 4: Modified BFV Encryption

Lemma 3.1 *Let $\texttt{ct}'_0$ be the output of Algorithm 4. Suppose that $\|s\|_\infty = 1$, $t|(p_0-1)$, $t|(q-1)$, and $p_0 > \frac{4\delta_R^2+2\delta_R}{\delta_R-1}$. Then $\texttt{ct}'_0$ is a BFV ciphertext with noise bounded by ρ.*

We argue that when $\delta_R \geq 16$, the condition on p_0 in Lemma 3.1 is satisfied when p_0 is chosen so that $p_0 \geq 5\delta_R + 3$ as per our parameter specifications, since

$$5\delta_R + 3 > \frac{32}{7}\delta_R + \frac{16}{7} = \frac{16}{7}(2\delta_R + 1) = \frac{2\delta_R(2\delta_R+1)}{\frac{7}{8}\delta_R} \geq \frac{4\delta_R^2 + 2\delta_R}{\delta_R - 1}.$$

This technique of encryption with a built-in modulus reduction in Step 3 was first mentioned in [19], but is overall not especially well outlined in the literature. Implementations do often reduce the modulus immediately after encryption to reduce noise. For instance, Microsoft SEAL [27] chooses p_0 as a "special prime", then generates all keys with an integer modulus of p_0q (for q a product of some primes) before reducing down to integer modulus q to house any ciphertext data. SEAL documentation recommends choosing this special prime p_0 to be at least as big as any prime divisor of q, though it is not a strict requirement. In our modification, we propose computing the modulus reduction locally during encryption, and doing so before adding the message bits. The advantage to this approach is we can choose p_0 to be much smaller.

Algorithm 5 provides for the decryption of a BFV ciphertext, which is the standard BFV decryption.

Lemma 3.2 *If the input $\texttt{ct}_0$ of Algorithm 5 is a BFV ciphertext with noise bounded by $(D_q - 1)/2$ and $t|(q-1)$, then the decryption in Algorithm 5 is correct.*

	$\texttt{BFV.Decrypt}(\texttt{ct}_0, \texttt{sk})$
Input:	$\texttt{ct}_0 = (a_0, b_0) \in R_{n,q}^2$ BFV ciphertext,
	$\texttt{sk} = s \in R_{n,3}$ secret key.
Output:	$m_0 \in R_{n,t}$ message.
Step 1.	Compute $c := [b_0 + a_0 s]_{\phi(x),q}$.
Step 2.	Compute $m_0 := \left[\left\lfloor \frac{tc}{q} \right\rceil\right]_t$.
Step 3.	Return m_0.

Algorithm 5: BFV Decryption

BFV Additions and Linear Combinations. We allow for linear combinations of ciphertexts with scalars from $\mathbb{Z}$. Based on the sum of absolute values of these scalars, we can guarantee a bound on the resulting ciphertext noise. In particular, we discuss the case of linear combinations with scalars $\alpha_0, \ldots, \alpha_{k-1} \in \mathbb{Z}$ such that $\sum_{i=0}^{k-1} |\alpha_i| \leq M$. Assuming each input ciphertext has noise bounded by E, a linear combination of k ciphertexts using these scalars results in noise bounded by $M(E+1)$. Algorithm 6 gives the algorithm for linear combinations, while Lemma 3.3 gives the resulting noise bound for BFV ciphertexts.

	$\texttt{Linearcombo}(\texttt{ct}_0, \ldots, \texttt{ct}_{k-1}, \alpha_0, \ldots, \alpha_{k-1})$
Input:	$\texttt{ct}_0, \ldots, \texttt{ct}_{k-1} \in R_{n,q}^2$ (or $\texttt{ct}_0, \ldots, \texttt{ct}_{k-1} \in R_{n,q}^3$),
	$\alpha_0, \ldots, \alpha_{k-1} \in \mathbb{Z}$ scalars.
Output:	$\texttt{ct}'_0 \in R_{n,q}^2$ (or $\texttt{ct}'_0 \in R_{n,q}^3$).
Step 1.	Set $\texttt{ct}'_0 := [0, 0]$ (or $[0, 0, 0]$).
	For i from 0 to $k-1$ do
	$\quad \texttt{ct}'_0 := [\texttt{ct}'_0 + \alpha_i \texttt{ct}_i]_q$.
Step 2.	Return $\texttt{ct}'_0$.

Algorithm 6: Linear Combinations

Lemma 3.3 *Suppose the inputs of Algorithm 6 are BFV ciphertexts each with noise bounded by E and suppose $\sum_{i=0}^{k-1} |\alpha_i| \leq M$. Let $\texttt{ct}'_0$ be the output of Algorithm 6. If $t|(q-1)$, then $\texttt{ct}'_0$ is a BFV ciphertext with noise bounded by $M(E+1)$.*

We remark that we allow for inputs of Algorithm 6 to also be in $R_{n,q}^3$. For the input of Algorithm 6, when using elements of the form $(c_0, c_1, c_2) \in R_{n,q}^3$ satisfying

$$c_0 + c_1 s + c_2 s^2 \equiv D_q m + e \mod (\phi(x), q)$$

for some $m \in R_{n,t}$ and $e \in R_n$, we still refer to e as a noise term (and refer to a bound on $\|e\|_\infty$ as a noise bound). If each input has noise bounded by E, we can slightly adjust the proof of Lemma 3.3 to obtain an element in $R^3_{n,q}$ with noise bounded by $M(E+1)$ from the output of Algorithm 6. This alternate choice of inputs will be utilized when discussing budgeted operations in Section 4.1.

BFV Multiplication. As expected, multiplication incurs much bigger increase in ciphertext noise and more tedious noise analysis. The procedure is again standard for the BFV scheme as in [6]. The proof is similar to [6], but we give a simpler worst case noise bound with Lemma 3.4.

	$\texttt{BFV.Multiply}(\texttt{ct}_0, \texttt{ct}_1)$
Input:	$\texttt{ct}_0 = (a_0, b_0), \texttt{ct}_1 = (a_1, b_1) \in R^2_{n,q}$ BFV ciphertexts.
Output:	$(c'_0, c'_1, c'_2) \in R^3_{n,q}$.
Step 1.	Compute
	$c_0 := [b_0 b_1]_{\phi(x)},\ c_1 := [b_1 a_0 + b_0 a_1]_{\phi(x)},\ c_2 := [a_0 a_1]_{\phi(x)}$.
Step 2.	Compute
	$c'_0 := \lfloor tc_0/q \rceil, \quad c'_1 := \lfloor tc_1/q \rceil, \quad c'_2 := \lfloor tc_2/q \rceil$.
Step 3.	Return (c'_0, c'_1, c'_2).

Algorithm 7: BFV Multiplication

Lemma 3.4 *Suppose the inputs of Algorithm 7 are BFV ciphertexts for messages m_0 and m_1 respectively, both with noise bounded by E. Let (c'_0, c'_1, c'_2) be the output of Algorithm 7. If $t|(q-1)$ and $\delta_R \geq 16$, then*

$$c'_0 + c'_1 s + c'_2 s^2 \equiv D_q[m_0 m_1]_{\phi(x),t} + e' \mod (\phi(x), q) \tag{1}$$

with $\|e'\|_\infty \leq 3.5Et\rho^2$.

The simple bound provided in Lemma 3.4 will allow us to easier choose parameters and stack moduli as we do in Section 4, while having minimal influence on functionality.

Comparison to current bounds. From the above, we can see that our bound is on the order of $Et\delta_R^2$ when choosing $\|s\|_\infty$=1, where E is the bound on the noise term of each ciphertext before multiplication. Comparing to more current works, [16] achieves a similar multiplication noise bound. Rather than restricting choices of t and q, [16] achieves this bound by an alternative encryption method, namely by computing $b_0 + a_0 s \equiv \lfloor qm/t \rceil + e \mod (\phi(x), q)$ rather than standard BFV, which computes $b_0 + a_0 s \equiv \lfloor q/t \rfloor m + e \mod (\phi(x), q)$. Below we provide a short comparison

of multiplication noise bounds, with e' being the noise term resulting from multiplication. Most notably, we assume two ciphertext noise terms are bounded both by E rather than having separate input bounds. Note this comparison does not include relinearization noise. We discuss the additional relinearization noise accumulated for our modified BFV scheme in Lemma 3.5.

Classic BFV [6]:

$$\left\|e'\right\|_\infty \leq 2\delta_R tE(1+\delta_R \left\|s\right\|_\infty) + 2\delta_R^2 t^2(\left\|s\right\|_\infty + 1)^2$$

Improved BFV [16]:

$$\left\|e'\right\|_\infty \leq \frac{\delta_R t}{2}\left(\frac{2E}{q} + (4+\delta_R \left\|s\right\|_\infty)2E\right) + \frac{1+\delta_R \left\|s\right\|_\infty + \delta_R^2 \left\|s\right\|_\infty^2}{2}$$

Our BFV Variant:

$$\left\|e'\right\|_\infty \leq 3.5Et\delta_R^2 \left\|s\right\|_\infty^2$$

In [16], the dominant noise term is $\frac{\delta_R t}{2}(\delta_R \left\|s\right\|_\infty)2E = Et\delta_R^2 \left\|s\right\|_\infty$. We note that, by going through their proof for the worst case bound, the factor $\delta_R/2$ in their bound should be $\delta_R \left\|s\right\|_\infty$, which is what we used in our proof. Hence the dominant term should be $2Et\delta_R^2 \left\|s\right\|_\infty$. In comparison, our bound for all the noise terms is $3.5Et\delta_R^2 \left\|s\right\|_\infty^2$, which is slightly bigger than their bound. Our goal was to provide a simple bound that is easier to use in practice. We'll expand upon how we use this simple bound further in Section 4.

BFV Relinearization. In order to convert a returned ciphertext from Algorithm 7 back to the proper form of a BFV ciphertext, we can employ a relinearization (or keyswitch) algorithm [6]. The algorithm converts a linear form in s and s^2 to a linear form in only s, while introducing a small additional noise term. Note that in order to accomplish this, we must use the published *evaluation key*, from Algorithm 3, denoted `ek`. Algorithm 8 gives the relinearization algorithm for BFV. Lemma 3.5 provides for the resulting noise bound.

Lemma 3.5 *Let (c_0, c_1) be the output of Algorithm 8 and suppose the input (c_0', c_1', c_2') satisfies (1) in Lemma 3.4. If $p_1 \geq 6q$ and $\delta_R \geq 16$, then (c_0, c_1) is a BFV ciphertext with noise bounded by $3.6Et\rho^2$.*

Alternate Relinearization Technique. Algorithm 8 is not the only option for relinearizing a ciphertext. Another technique [6, 14, 25, 28] involves generating the evaluation key differently, by expanding c_2' with respect to some integer base. In this relinearization process, the evaluation key is a vector pair $\texttt{ek} = (\mathbf{u}, \mathbf{v}) \in (R_{n,q}^\gamma)^2$ where each entry of $\mathbf{u}$ is sampled from $U(R_{n,q})$, and γ and $\mathbf{v}$ are computed in the following

	$\texttt{BFV.Relinearize}((c'_0, c'_1, c'_2), \texttt{ek})$
Input:	$(c'_0, c'_1, c'_2) \in R^3_{n,q}$ polynomial ordered triple,
	$\texttt{ek} = (\texttt{k}'_0, \texttt{k}'_1) \in R^2_{n,p_1 q}$ evaluation key.
Output:	$(c_0, c_1) \in R^2_{n,q}$.
Step 1.	Compute $\beta_0 := [c'_2 \texttt{k}'_0]_{\phi(x),p_1 q}$ and $\beta_1 := [c'_2 \texttt{k}'_1]_{\phi(x),p_1 q}$.
Step 2.	Compute $d'_0 := \left\lfloor \frac{\beta_0}{p_1} \right\rceil$ and $d'_1 := \left\lfloor \frac{\beta_1}{p_1} \right\rceil$.
Step 3.	Compute $c_0 := [c'_0 + d'_0]_q$ and $c_1 := [c'_1 + d'_1]_q$.
Step 4.	Return (c_0, c_1).

Algorithm 8: BFV Relinearization

way. For chosen public base $B \in \mathbb{N}$, find the smallest $\gamma \in \mathbb{N}$ such that $B^\gamma > q$ and define $\mathbf{g} \in R^\gamma_{n,q}$ as

$$\mathbf{g}^T = \left(1, B, B^2, \ldots, B^{\gamma-1}\right).$$

Let $\mathbf{w} \in R^\gamma_{n,q}$ a vector with each entry sampled from χ_ρ. Compute $\mathbf{v}$ as

$$\mathbf{v} := s^2\mathbf{g} - \mathbf{u}s + \mathbf{w} \pmod{\phi(x), q}.$$

To obtain a new relinearized ciphertext from (c'_0, c'_1, c'_2), one can first write

$$c'_2 = \sum_{j=0}^{\gamma-1} h_j B^j$$

where $h_j \in R_{n,q}$ such that $\left\|h_j\right\|_\infty \leq B/2$ and define $\mathbf{h}^T \in R^\gamma_{n,q}$ as $\mathbf{h} = \left(h_0, h_1, \ldots, h_{\gamma-1}\right)$. Then

$$c'_2 s^2 = (\mathbf{hg})s^2 = \left(\sum_{j=0}^{\gamma-1} h_j B^j\right) s^2.$$

Using $\texttt{ek} = (\mathbf{u}, \mathbf{v})$, the new ciphertext can be computed as $([c'_1 + \mathbf{hu}]_{\phi(x),q}, [c'_0 + \mathbf{hv}]_{\phi(x),q})$ since $c'_2 s^2 + \mathbf{hw} \equiv \mathbf{hv} + \mathbf{hu}s \mod (\phi(x), q)$. Here, $\mathbf{hw}$ is the noise introduced during relinearization and satisfies $\|\mathbf{hw}\|_\infty \leq (\gamma B \delta_R^2 \|s\|_\infty)/2$. However, this technique is less used since the evaluation key $(\mathbf{u}, \mathbf{v}) \in (R^\gamma_{n,q})^2$ is much larger than the evaluation key generated in Algorithm 3. Specifically, $\texttt{ek} = (\mathbf{u}, \mathbf{v})$ is of size $2\gamma \log_2 q$. The noise incurred by relinearization grows linearly with B, hence B must be relatively small, which means γ will likely be much bigger than 4. On the other hand, $\texttt{ek} = (\texttt{k}'_0, \texttt{k}'_1)$ from Algorithm 3 is of size $4 \log_2 q$. Although Algorithm 3 gives a smaller key size, a larger ring dimension n must be used to maintain security. We refer the reader to the references above for details. Some implementations of BFV such as Microsoft SEAL [27] do not realinearize their ciphertexts after each multiplication and allow the degree of the linear form s to grow larger than 2 [29].

3.2 Modified BGV Scheme

BGV Key Generation. As we did with BFV, we use a slightly different key generation process from the standard BGV scheme [4, 5] by generating the public key and evaluation key in a larger modulus to reduce noise sizes in ciphertexts. Algorithm 9 gives the key generation for the BGV keys. Just like BFV, $\mathtt{sk}$ is kept secret, while $\mathtt{pk}$ and $\mathtt{ek}$ are published.

	$\mathtt{BGV.Keygen}(q, p_0, p_1)$
Input:	$q \in \mathbb{N}$,
	$p_0 \in \mathbb{N}$ with $p_0 \geq 5\delta_R + 3$,
	$p_1 \in \mathbb{N}$ with $p_1 \geq 6q$.
Output:	$\mathtt{sk} = s \in R_{n,3}$ secret key,
	$\mathtt{pk} = (\mathtt{k}_0, \mathtt{k}_1) \in R^2_{n,p_0q}$ public key,
	$\mathtt{ek} = (\mathtt{k}'_0, \mathtt{k}'_1) \in R^2_{n,p_1q}$ evaluation key.
Step 1.	Choose randomly $s \in R_{n,3}$.
Step 2.	Sample $\mathtt{k}_0 \leftarrow U(R_{n,p_0q})$ and $e \leftarrow \chi_\rho$.
	Compute $\mathtt{k}_1 := [-(\mathtt{k}_0 s + te)]_{\phi(x),p_0q}$.
Step 3.	Sample $\mathtt{k}'_1 \leftarrow U(R_{n,p_1q})$ and $e'_1 \leftarrow \chi_\rho$.
	Compute $\mathtt{k}'_0 := [-\mathtt{k}'_1 s + p_1 s^2 + te'_1]_{\phi(x),p_1q}$.
Step 4.	Return $\mathtt{sk} = s$, $\mathtt{pk} = (\mathtt{k}_0, \mathtt{k}_1)$, and $\mathtt{ek} = (\mathtt{k}'_0, \mathtt{k}'_1)$.

Algorithm 9: BGV Key Generation

BGV Encryption and Decryption. We define the BGV public key encryption in Algorithm 10. Decryption of a BGV ciphertext is given in Algorithm 11. When we refer to a "BGV ciphertext" in these algorithms and lemmas, we mean an ordered pair $\mathtt{ct} = (a, b) \in R^2_{n,q}$ satisfying

$$b + as \equiv m + te \mod (\phi(x), q)$$

for some noise term $e \in R_n$ and given message $m \in R_{n,t}$. Lemma 3.6 provides provides for proof of correctness, as well as the corresponding noise bound resulting from encryption.

Lemma 3.6 *Let $\mathtt{ct}'_0$ be the output of Algorithm 10. Suppose that $\|s\|_\infty = 1$, $t|(p_0-1)$, $t|(q-1)$, and $p_0 > \frac{4\delta_R^2+2\delta_R}{\delta_R-2}$. Then $\mathtt{ct}'_0$ is a BGV ciphertext with noise bounded by ρ.*

Just as with BFV, the condition on p_0 can be discussed in the more general case for any choice of u and s, in which the condition on p_0 is $p_0 > \frac{2\delta_R^2(\|u\|_\infty+\|s\|_\infty)+2\delta_R}{\delta_R\|s\|_\infty-2}$ and

	$\texttt{BGV.Encrypt}(m_0, \texttt{pk})$
Input:	$m_0 \in R_{n,t}$ message,
	$\texttt{pk} = (\texttt{k}_0, \texttt{k}_1) \in R^2_{n,p_0q}$ public key,
Output:	$\texttt{ct}'_0 = (a'_0, b'_0) \in R^2_{n,q}$ BGV ciphertext.
Step 1.	Sample $u \leftarrow U(R_{n,3})$, and sample $e_1, e_2 \leftarrow \chi_\rho$.
Step 2.	Compute $(a_0, b_0) \in R^2_{n,p_0q}$ where
	$a_0 := [\texttt{k}_0 u + te_1]_{\phi(x),p_0q}$,
	$b_0 := [\texttt{k}_1 u + te_2]_{\phi(x),p_0q}$.
Step 3.	Compute
	$(a'_0, b^*_0) := \texttt{BGV.Modreduce}((a_0, b_0), p_0q, q)$,
	$b'_0 := [b^*_0 + m_0]_q$.
Step 4.	Return $\texttt{ct}'_0 = (a'_0, b'_0) \in R^2_{n,q}$.

Algorithm 10: Modified BGV Encryption

	$\texttt{BGV.Decrypt}(\texttt{ct}_0, \texttt{sk})$
Input:	$\texttt{ct}_0 = (a_0, b_0) \in R^2_{n,q}$ BGV ciphertext,
	$\texttt{sk} = s \in R_{n,3}$ secret key.
Output:	$m_0 \in R_{n,t}$ message.
Step 1.	Compute $c := [b_0 + a_0 s]_{\phi(x),q}$.
Step 2.	Compute $m_0 := [c]_t$.
Step 3.	Return m_0.

Algorithm 11: BGV Decryption

the resulting noise term e'_0 satisfies $\left\|e'_0\right\| < \rho$. Similar to BFV as well, we argue that when $\delta_R \geq 16$, the condition on p_0 in Lemma 3.6 is satisfied when p_0 is chosen so that $p_0 \geq 5\delta_R + 3$ as per our parameter specifications, since

$$5\delta_R + 3 > \frac{32}{7}\delta_R + \frac{16}{7} = \frac{16}{7}(2\delta_R + 1) = \frac{2\delta_R(2\delta_R + 1)}{\frac{7}{8}\delta_R} \geq \frac{4\delta_R^2 + 2\delta_R}{\delta_R - 2}.$$

Regarding decryption, the proof of correctness for Algorithm 11 is straightforward. Simply observe that $[[b_0 + a_0 s]_{\phi(x),q}]_t = [m_0 + te_0]_t = m_0$. The key observation here is that in order for correctness to hold, it is required that $\|m_0 + te_0\|_\infty < q/2$. That is, fully reducing $b_0 + a_0 s$ modulo q will actually yield the correct polynomial $m_0 + te_0$. The worst case bound on $\|m_0 + te_0\|_\infty$ is $t/2 + tE$ if $\|e_0\|_\infty \leq E$. Hence, it suffices

to require $E < \frac{q}{2t} - \frac{1}{2}$. This is very similar to the condition given earlier needed for correct BFV decryption.

BGV Additions and Linear Combinations. Additions and linear combinations for BGV can be done using Algorithm 6. The argument is similar to Lemma 3.3 for BFV ciphertexts, and results in the same noise bound of $M(E+1)$. It is worth noting that divisibility of $q-1$ by t yields no noise improvement for BGV addition, and the noise bound of $M(E+1)$ holds for any t and q.

BGV Multiplication. Again, multiplication incurs large noise increase during homomorphic computation. Unlike BFV, there is no requirement that t divide $q-1$, and he noise analysis for BGV multiplication is simpler than BFV, as no scaling by t/q is required after computing the necessary components given from $\mathtt{ct}_0$ and $\mathtt{ct}_1$. Algorithm 12 outlines the procedure for BGV multiplication. Lemma 12 provides for proof of correctness and the corresponding noise bound.

	$\mathtt{BGV.Multiply}(\mathtt{ct}_0, \mathtt{ct}_1)$
Input:	$\mathtt{ct}_0 = (a_0, b_0), \mathtt{ct}_1 = (a_1, b_1) \in R^2_{n,q}$ ciphertexts.
Output:	$(c'_0, c'_1, c'_2) \in R^3_{n,q}$.
Step 1.	Compute $c'_0 := [b_0 b_1]_{\phi(x),q}$, $c'_1 := [b_1 a_0 + b_0 a_1]_{\phi(x),q}$, $c'_2 := [a_0 a_1]_{\phi(x),q}$.
Step 2.	Return (c'_0, c'_1, c'_2).

Algorithm 12: BGV Multiplication

Lemma 3.7 *Suppose the inputs of Algorithm 12 are BGV ciphertexts for messages m_0 and m_1 respectively, both with noise bounded by E. Let (c'_0, c'_1, c'_2) be the output of Algorithm 12. Then*

$$c'_0 + c'_1 s + c'_2 s^2 \equiv [m_0 m_1]_{\phi(x),t} + te' \mod (\phi(x), q) \tag{2}$$

with $\left\|e'\right\|_\infty \leq 2\delta_R t(E^2+1)$.

BGV Relinearization. We can relinearize a BGV ciphertext to rewrite the left hand side of equation 2 as a linear form in only s rather than s and s^2. For BGV, a slightly modified evaluation key must be generated, as well as a slightly modified relinearization algorithm. Algorithms 9 and 13 give the BGV evaluation key generation and relinearization respectively, which we've based on the algorithms in [16]. Lemma 3.8 provides proof of correctness of Algorithm 13 and the corresponding noise bound. Algorithm 13 and the result of Lemma 3.8 can be combined with Algorithm 12 and the result of Lemma 12, respectively, to obtain a full BGV multiplication operation.

	$\texttt{BGV.Relinearize}((c'_0, c'_1, c'_2), \texttt{ek})$
Input:	$(c'_0, c'_1, c'_2) \in R^3_{n,q}$,
	$\texttt{ek} = (\texttt{k}'_0, \texttt{k}'_1) \in R^2_{n,p_1 q}$ evaluation key.
Output:	$(c_0, c_1) \in R^2_{n,q}$.
Step 1.	Compute $\beta_0 := [c'_2 \texttt{k}'_0]_{\phi(x),p_1 q}$ and $\beta_1 := [c'_2 \texttt{k}'_1]_{\phi(x),p_1 q}$.
Step 2.	Compute $\omega_0 := [-t^{-1}\beta_0]_{p_1}$ and $\omega_1 := [-t^{-1}\beta_1]_{p_1}$.
Step 3.	Compute $d'_0 := \frac{\beta_0 + t\omega_0}{p_1}$ and $d'_1 := \frac{\beta_1 + t\omega_1}{p_1}$.
Step 4.	Compute $c_0 := [c'_0 + d'_0]_q$ and $c_1 := [c'_1 + d'_1]_q$.
Step 5.	Return (c_0, c_1).

Algorithm 13: BGV Relinearization

Lemma 3.8 *Let (c_0, c_1) be the output of Algorithm 13 and suppose the input (c'_0, c'_1, c'_2) satisfies (2) in Lemma 3.7. If $p_1 \geq 6q$ and $\delta_R \geq 16$, then (c_0, c_1) is a BGV ciphertext with noise bounded by $2\delta_R t(E^2 + 1) + \frac{1}{8}\delta_R^2 \|s\|_\infty$.*

3.3 Modified CKKS Scheme

In this section, we'll discuss the CKKS scheme [7]. CKKS allows for homomorphic encryption for arithmetic of approximate numbers rather than arithmetic exactly as BFV and BGV do. This is done by first taking in data as some vector over $\mathbb{C}$, mapping the components into R_n, and then performing the homomorphic computation before mapping back to a vector over $\mathbb{C}$. This process of mapping to and from the $\mathbb{C}$-vector space is known as the *encoding* and *decoding* procedures, respectively. Throughout Section 3.3 and whenever referring to CKKS, we will always assume $\phi(x) = x^n + 1$ where n is a power of two. Thus, $\delta_R = n$.

Message Encoding and Decoding. Recall that $R_n = \mathbb{Z}[x]/(\phi(x))$ and $R_{n,q} = \mathbb{Z}_q[x]/(\phi(x))$. Let $\mathbb{H} = \{z \in \mathbb{C}^n : z_j = \overline{z_{n-j}}\}$. Define two mappings:

$$\pi : \mathbb{H} \to \mathbb{C}^{n/2},$$
$$\sigma : \mathbb{C}[x]/(\phi(x)) \to \mathbb{C}^n$$

as follows. Here, π is the projection of $\mathbb{H}$ onto $\mathbb{C}^{n/2}$, by keeping only the first half of the entries for each vector in $\mathbb{H}$, and σ is the canonical embedding map defined as follows. Note that the polynomial $\phi(x) = x^n + 1$ has n complex roots, say $\zeta_1, \zeta_2, \ldots, \zeta_n$ in any fixed order, which are all primitive roots of unity of order $2n$. Given a polynomial $h \in \mathbb{C}[x]/(\phi(x))$, σ is defined via

$$\sigma(h) = (h(\zeta_1), h(\zeta_2), \ldots, h(\zeta_n)) \in \mathbb{C}^n.$$

That is, σ evaluates h at all the roots of $\phi(x)$ and stores the evaluations as a vector. Note that π and σ both serve as isomorphisms of vector spaces over $\mathbb{C}$, so π^{-1} and σ^{-1} exist. In practice, σ is computed via a fast Fourier transform (FFT), and σ^{-1} by an inverse fast Fourier transform (FFT^{-1}).

The purpose of these mappings is that given a message vector $z \in \mathbb{C}^{n/2}$, we want to convert it into a polynomial in R_n whose values at ζ_i correspond to z, hence polynomial multiplication corresponds to component-wise multiplication for message vectors. Given $z \in \mathbb{C}^{n/2}$, computing $\pi^{-1}(z)$ is easy. We now must map $\pi^{-1}(z)$ into R_n. Given $\zeta_1, \zeta_2, \ldots, \zeta_n$ in a fixed order such that $(\zeta_1, \zeta_2, \ldots, \zeta_n) \in \mathbb{H}$, σ then serves as an isomorphism between $\mathbb{R}[x]/(\phi(x))$ and $\mathbb{H}$. So, for $w \in \mathbb{H}$, we can compute $\sigma^{-1}(w) \in \mathbb{R}[x]/(\phi(x))$, and then round each coefficient to obtain an element in R_n.

It is worth noting that most texts use a technique called *coordinate-wise random rounding* instead of rounding to the nearest integer [24]. However, we will use the closest integer rounding in this paper. As we'll see, this step of rounding causes accuracy loss in the message. To avoid this, we scale by some positive integer Δ in order to preserve some desired precision of our message in the end result. The message encoding function is defined as

$$\texttt{Ecd}(z, \Delta) = \lfloor \sigma^{-1}(\Delta \pi^{-1}(z)) \rceil \in R_n,$$

for any message $z \in \mathbb{C}^{n/2}$, and the message decoding function is defined as

$$\texttt{Dcd}(m, \Delta) = \pi(\sigma(\Delta^{-1} m)),$$

for any polynomial $m \in R_n$.

The encryption and decryption procedures for CKKS then map between R_n and $R_{n,q}$. A high level overview of the mappings in CKKS is shown below. Note that q' is used for the integer modulus of the ciphertext space after homomorphic computation, as we may have a different integer modulus if we perform any modulus reduction.

Overview of CKKS Mappings

$$\begin{array}{ccccccccc}
\mathbb{C}^{n/2} & \xrightarrow{\Delta\pi^{-1}} & \mathbb{H} & \xrightarrow{\sigma^{-1}} & \mathbb{R}[x]/(\phi(x)) & \xrightarrow{\lfloor\cdot\rceil} & R_n & \xrightarrow{\texttt{encrypt}} & R_{n,q}^2 \\
 & & & & & & & & \Big\downarrow \texttt{computation} \\
\mathbb{C}^{n/2} & \xleftarrow[\Delta^{-1}\pi]{} & \mathbb{H} & \xleftarrow{\qquad\sigma\qquad} & & & R_n & \xleftarrow[\texttt{decrypt}]{} & R_{n,q'}^2
\end{array}$$

Note the scaling factor Δ affects the ending precision and is usually chosen proportionally to the moduli gaps, which is discussed later in this section. It is also worth mentioning that although `Ecd` is defined for all messages $z \in \mathbb{C}^{n/2}$, in practice z is taken in the space of fixed precision numbers of some length, which is a subset of $\mathbb{C}^{n/2}$.

The remainder of Section 3.3 is devoted to the homomorphic computation in $R_{n,q}^2$ for the CKKS scheme. A significant observation for CKKS is how the homomorphic

computation relates to the computation in $\mathbb{C}^{n/2}$. In particular, for vectors $z, z' \in \mathbb{C}^{n/2}$, we denote $z \circ z'$ as the Hadamard product of z and z' (i.e., the vector obtained from component-wise multiplication between z and z'). Homomorphic multiplication in $R_{n,q}^2$ of two ciphertexts corresponds with the Hadamard product of the respective vectors in $\mathbb{C}^{n/2}$, whereas homomorphic addition corresponds with standard vector addition of the respective message vectors.

CKKS Rescaling. Regarding modulus reduction in CKKS, a similar procedure known as *rescaling* occurs. The rescaling procedure is identical to the modulus reduction for BFV outlined in Algorithm 1. That is,

$$\texttt{CKKS.Modreduce} = \texttt{BFV.Modreduce}.$$

The main difference is the purpose of the procedure. Rather than using modulus reduction as a form of noise control, it is used here to control precision. For two message encodings $m_0, m_1 \in R_n$, ciphertext multiplication yields an encryption of the product $m_0 m_1$, which takes up some less significant bits (LSBs). We rescale the corresponding ciphertext of $m_0 m_1$ to get rid of the lower significant digits in order to perform further computation where we want to keep only fixed number of digits. While the rescaling serves a different purpose than modulus reduction, we can still discuss bounds on the corresponding error term achieved. Lemma 3.9 outlines our worst case noise bound. When we refer to a "CKKS ciphertext" in these algorithms and lemmas, we mean an ordered pair $\texttt{ct} = (a, b) \in R_{n,q}^2$ satisfying

$$b + as \equiv m + e \mod (\phi(x), q)$$

for some noise term $e \in \mathbb{R}[x]/(\phi(x), q)$ and $m = \texttt{Ecd}(z, \Delta) \in \mathbb{R}[x]/(\phi(x), q)$ for some $z \in \mathbb{C}^{n/2}$.

Lemma 3.9 *Suppose the input of Algorithm 1 is a CKKS ciphertext with noise bounded by E. Let $\texttt{ct}_0'$ be the output of Algorithm 1. Then,*

$$b_0' + a_0' s \equiv \frac{q}{Q} m_0 + e_0' \mod (\phi(x), q)$$

and $\left\|e_0'\right\|_\infty \leq \frac{q}{Q} E + \frac{1+\delta_R \|s\|_\infty}{2}$. Furthermore, if $Q/q > \frac{2E}{\delta_R \|s\|_\infty - 1}$, then $\left\|e_0'\right\|_\infty < \rho$.

A notable difference in CKKS rescaling is that the algorithm returns an encryption of $\frac{q}{Q} m_0$ rather than the original m_0 encoding. As mentioned, this is intentional, as we wish to reduce the size of m_0 since bit usage becomes an issue. The reason we use a modulus reduction algorithm rather than simply trying to scale down the ciphertext is because we are taking entries modulo Q. For a ciphertext $(a_0, b_0) \in R_{n,Q}^2$, if we computed a scaled ciphertext $(\lfloor a_0/\Delta \rceil, \lfloor b_0/\Delta \rceil)$ for some scaling factor Δ, we would first need to write $b_0 + a_0 s \equiv m_0 + e_0 + Qr$ for a polynomial $r \in R_n$. This would result in a term approximately equal to Qr/Δ after dividing through by Δ, which is

no longer equivalent to 0 mod Q and would result in a huge noise term. However, it is still important to choose Q/q to be approximately Δ, or whatever desired scaling factor is needed. Accuracy of the approximation relies on this size of Q/q. When not concerned with RNS representation, we can simply choose $\Delta = Q/q$ exactly. In the RNS variant of CKKS [21], a bound is placed on the gap between Q/q and Δ to ensure some precision, while still allowing for coprime moduli Q and q.

CKKS Key Generation. For CKKS, the keys used are generated in the same way that the BFV keys are generated. In this case, we refer the reader back to Algorithm 3 for generation of the CKKS keys, which again includes the secret key `sk`, the public key `pk`, and the evaluation key `ek`.

CKKS Encryption and Decryption. The encryption algorithm is given by Algorithm 14, and decryption by Algorithm 15. Note that CKKS encryption in Algorithm 14 uses Algorithm 1 as a subroutine, which is the rescaling. From step 2 of Algorithm 15, we obtain m_0'. However, recall that $b_0 + a_0 s \equiv m_0 + e_0 \mod (\phi(x), q)$, so in this case we really have that $m_0' = m_0 + e_0$. In other words, m_0' is a close approximation of m_0 so long as $\|e_0\|_\infty$ is small. We do not include proofs that decryption works, as it is directly apparent from the algorithm that it decrypts to an approximation of the desired message. We also note that many texts, such as the original CKKS paper in [7], have separate steps for encoding/decoding and encryption/decryption. We however, include the encoding or decoding in the encryption or decryption algorithms respectively. Aside from the encoding step, the encryption algorithm for CKKS is actually identical to a BFV encryption with $D_q = 1$. Lemma 3.10 provides for the corresponding noise bound after encryption. As with the other schemes, choosing $p_0 \geq 5\delta_R + 3$ ensures the condition for p_0 in Lemma 3.10 holds. The condition on p_0 can also be generalized to $p_0 > \frac{2\delta_R^2(\|u\|_\infty + \|s\|_\infty) + 2\delta_R}{\delta_R \|s\|_\infty - 1}$ with the resulting noise still satisfying $\left\|e_0'\right\| < \rho$.

	$\texttt{CKKS.Encrypt}(z, \Delta, \texttt{pk})$
Input:	$z \in \mathbb{C}^{n/2}$ message,
	$\Delta \in \mathbb{N}$ scaling factor,
	$\texttt{pk} = (\texttt{k}_0, \texttt{k}_1) \in R^2_{n,p_0q}$ public key.
Output:	$\texttt{ct} = (a, b) \in R^2_{n,q}$ CKKS ciphertext.
Step 1.	Encode z by computing $m := \texttt{Ecd}(z, \Delta)$.
Step 2.	Compute $\texttt{ct} := \texttt{BFV.Encrypt}(m, 1, \texttt{pk})$.
Step 3.	Return $\texttt{ct} = (a, b) \in R^2_{n,q}$.

Algorithm 14: Modified CKKS Encryption

	$\texttt{CKKS.Decrypt}(\texttt{ct}, \texttt{sk})$
Input:	$\texttt{ct} = (a, b) \in R_{n,q}^2$ CKKS ciphertext,
	$\texttt{sk} = s \in R_{n,3}$ secret key.
Output:	$z \in \mathbb{C}^{n/2}$ message.
Step 1.	Compute $m := [b + as]_{\phi(x),q}$.
Step 2.	Decode m by computing $z := \texttt{Dcd}(m, \Delta)$.
Step 3.	Return z.

Algorithm 15: CKKS Decryption

Lemma 3.10 *Let* $\texttt{ct}$ *be the output of Algorithm* 14. *Suppose* $\|s\|_\infty = 1$ *and* $p_0 > \frac{4\delta_R^2 + 2\delta_R}{\delta_R - 1}$. *Then* $\texttt{ct}$ *is a CKKS ciphertext with noise bounded by* ρ.

CKKS Additions and Linear Combinations. We can perform additions and linear combinations with CKKS ciphertexts using Algorithm 6. The resulting ciphertext obtained from Algorithm 6 has a slightly different noise bound than BFV and BGV. The reason for this is that the encoded messages which the ciphertexts represent are in R_n instead of $R_{n,t}$, so reduction modulo t is not necessary with CKKS. The result is summarized in Lemma 3.11.

Lemma 3.11 *Suppose the inputs of Algorithm 6 are CKKS ciphertexts each with noise bounded by* E *and suppose* $\sum_{i=0}^{k-1} |\alpha_i| \leq M$. *Let* $\texttt{ct}_0'$ *be the output of Algorithm 6. Then,* $\texttt{ct}_0'$ *is a CKKS ciphertext with noise bounded by* ME.

CKKS Multiplication. Multiplication in CKKS follows the same process as BGV, which is given in Algorithm 12. Thus,

$$\texttt{CKKS.Multiply} = \texttt{BGV.Multiply}.$$

The difference is only a slightly different noise bound, due to the plaintexts not being in $R_{n,t}$ and thus not needing reduction modulo t. Lemma 3.12 outlines the result and proof of the corresponding noise bound.

Lemma 3.12 *Suppose the inputs of Algorithm 12 are CKKS ciphertexts for messages* m_0 *and* m_1, *respectively, both with noise bounded by* E. *Suppose that* $\|m_0\|_\infty \leq t/2$ *and* $\|m_1\|_\infty \leq t/2$. *Let* (c_0', c_1', c_2') *be the output of Algorithm* 12. *Then*

$$c_0' + c_1' s + c_2' s^2 \equiv m_0 m_1 + e' \mod (\phi(x), q) \tag{3}$$

with $\left\|e'\right\|_\infty \leq Et\delta_R + E^2\delta_R$.

CKKS Relinearization. As with the other schemes, full multiplication can then be achieved by including the relinearization process discussed in Algorithm 8, so

$$\texttt{CKKS.Relinearize} = \texttt{BFV.Relinearize}.$$

The proof is almost identical to the proof in Lemma 3.5. The result for CKKS is given below in Lemma 3.13.

Lemma 3.13 *Let (c_0, c_1) be the output of Algorithm 8 and suppose the input (c_0', c_1', c_2') satisfies (3) in Lemma 3.12, with $\|m_0\|_\infty \leq t/2$ and $\|m_1\|_\infty \leq t/2$. If $p_1 \geq 6q$ and $\delta_R \geq 16$, then (c_0, c_1) is a CKKS ciphertext with noise bounded by $Et\delta_R + E^2\delta_R + \frac{1}{8}\delta_R^2 \|s\|_\infty$.*

Note on BFV versus CKKS. Initially, the formatting of ciphertexts in BFV and CKKS seem very similar. For a message $m_0 \in R_{n,t}$, a BFV ciphertext $\texttt{ct}_0 = (a_0, b_0)$ satisfies $b_0 + a_0 s \equiv D_q m_0 + e_0 \mod (\phi(x), q)$. In CKKS, the encoding step for a message $z_0 \in \mathbb{C}^{n/2}$ scales our message by a factor of Δ. That is, our CKKS ciphertext $\texttt{ct}_0 = (a_0, b_0)$ satisfies $b_0 + a_0 s \equiv m_0 + e_0 \mod (\phi(x), q)$, where $m_0 = \texttt{Ecd}(z_0, \Delta)$. In both equations for BFV and CKKS, we have a scaling factor attached to our message. In BFV, the message m_0 is directly multiplied by D_q, while in CKKS Δ multiplies the original message z_0 and is implicitly hiding in m_0. Nonetheless, both have a scaling factor. In multiplication of both schemes, this scaling factor initially compounds in the first step. The two schemes handle this issue differently however, with BFV rescaling the individual degree 2 ciphertext components in step 2 of Algorithm 2, before taking the computed polynomials modulo q. CKKS on the other hand computes the initial polynomials in multiplication and immediately reduces them modulo q, relinearizes the ciphertext, and then rescales the hidden Δ^2 back to Δ using modulus reduction in Algorithm 1. Part of the reason these schemes differ in where they rescale is due to the fact that $D_q \gg \Delta$. Since $D_q = \lfloor q/t \rfloor$ and $t < q/2$, $D_q^2 > q$ and rescaling must occur before taking components modulo q. On the other hand, Δ in CKKS has more freedom in choice, as it is a parameter chosen by the user that can influence accuracy in the approximation of end result of computation.

3.4 Comparison to Other Noise Bound Analyses

Our noise analysis differs from previous works [6, 7, 16, 17] in that we derive *worst-case* bounds based on worst-case bound assumptions on the error distribution. As a result, our correctness guarantees are deterministic—there is no probability of decryption error. Additionally, we simplify the derived bounds into clean, closed-form expressions that will be useful for subsequent sections. This simplification comes at the cost of slightly looser bounds overall, with the effect being most pronounced in BFV multiplication. A detailed comparison of BFV multiplication appears near the end of Section 3.1.

For most operations, our bounds are very close to the worst-case results in [16], though with a few important distinctions. First, in all of our modulus reduction lemmas, we explicitly bound the ratio Q/q to ensure that the noise remains below a fixed threshold, which supports more precise composability in computations. Second, our modified encryption procedure enables much smaller choices of p_0 by performing modulus reduction *before* message embedding. This results in fresh ciphertexts with noise bounded by a constant, while preserving correctness. The same structure can be extended to CKKS rescaling during encryption. Since the message bits are not yet introduced at that stage, rescaling does not degrade precision.

In addition, under basic assumptions on δ_R, we are able to significantly simplify the relinearization noise bounds. This is especially helpful in regimes with small plaintext modulus t, where relinearization noise can be more significant.

For concrete estimates, many works adopt $\delta_R = 2\sqrt{n}$ as an expansion factor that holds with high probability. In contrast, we use the exact value $\delta_R = n$ in later derivations. In the case of CKKS [7, 17], the most significant differences in noise growth appear in fresh encryptions, relinearization (i.e., key switching), and rescaling. Across these operations, our analysis yields a dominant noise term of approximately n, compared to $\sqrt{n}$ in the aforementioned works. This is primarily due to our adoption of a strict worst-case model and the associated choice of expansion factor. For average-case analyses such as those in [17], direct comparison is more nuanced, as noise growth depends on the variance of sampled noise terms.

4 Leveled Schemes and RNS Variants

For practical computation, we employ a *leveled homomorphic encryption scheme* rather than a fully homomorphic one. Unlike fully homomorphic schemes—which support an unlimited number of operations via costly bootstrapping—a leveled scheme supports a predetermined number of homomorphic operations, making it more efficient for realistic workloads. In this section, we outline leveled versions of the BFV, BGV, and CKKS schemes. The core idea is to carry out computations at decreasing modulus levels: perform a fixed number of operations at a given modulus, then reduce both the modulus and the noise to enable further computation.

Let $q_\ell > q_{\ell-1} > \cdots > q_0 > 1$ be distinct primes, and define

$$Q_i = \prod_{j=0}^{i} q_j, \quad 0 \le i \le \ell.$$

We refer to Q_i as the *modulus at level* i or simply the *level-*i *modulus*. In Section 4.1, we describe how to select each q_i so that a budgeted operation, called a *depth-1 multiplication*, can be performed at level Q_i. Specifically, if the input ciphertexts at level Q_i have noise bounded by ρ, then the resulting ciphertext—after multiplication and modulus switching to Q_{i-1}—continues to maintain the same noise bound ρ.

Section 4.2 details how ciphertext operations are performed in the *residue number system* (RNS). By the Chinese Remainder Theorem, any polynomial $a \in R_{n,Q_\ell}$ can be represented as

$$[a]_{\mathcal{B}} = (a^{(0)}, a^{(1)}, \ldots, a^{(\ell)}),$$

where $a^{(i)} := a \bmod q_i$, and $\mathcal{B} = (q_0, q_1, \ldots, q_\ell)$ is called the *modulus basis* (or simply the *basis*). This representation is referred to as the RNS form of a.

All ciphertexts, public keys, and evaluation keys are stored in RNS form with respect to appropriate modulus bases. A key advantage of RNS is that addition and multiplication of polynomials can be performed *component-wise*, independently across the q_i. However, operations such as modulus reduction and relinearization are more involved. In Section 4.2, we describe how these operations are implemented in RNS for the BFV, BGV, and CKKS schemes, and we present the associated noise bounds.

4.1 Budgeted Operations at Each Level

For a collection of ciphertexts, we want to know how much homomorphic computation we can perform before ciphertext noise becomes too big so that no further computation can be performed. To do this, we introduce the concept of a depth-1 multiplication computation.

Definition 4.1 (Depth-1 Multiplication) *Suppose we have a collection of messages. For fixed k_1 and k_2, we say that we can perform a depth-1 multiplication if we can perform $2k_2$ groups of $k_1 - 1$ additions, followed by one round of k_2 multiplications, followed by $k_2 - 1$ additions.*

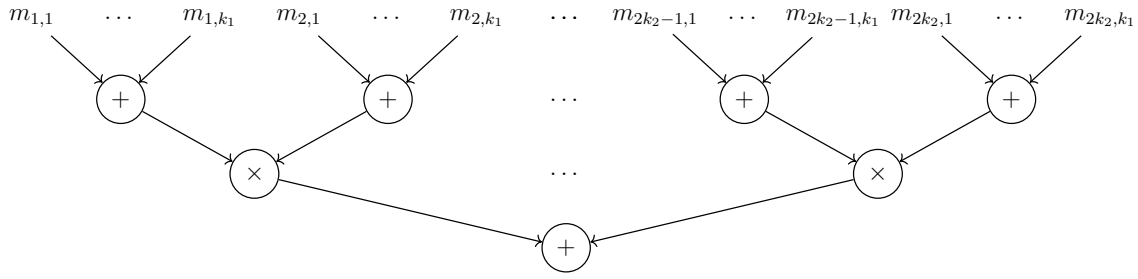

Figure 16: Plaintext Depth-1 Multiplication

Figure 16 shows an arbitrary depth-1 multiplication with $2k_2k_1$ plaintexts, where $m_{j,k}$ is a plaintext for each $j = 1, \ldots, 2k_2$, $k = 1, \ldots, k_1$. Our goal to derive a bound on q_i so that one can compute depth-1 multiplication homomorphically at each level i. To perform a depth-1 multiplication homomorphically for BFV, BGV, and CKKS, we introduce Algorithm 17.

In Algorithm 17, we remark that `Multiply`, `Relinearize`, and `Modreduce` call the respective algorithms for the inputted ciphertext type. For example, if each

	$\texttt{Depth1}(\texttt{ct}_{j,k}, \texttt{ek}_i, Q_i, Q_{i-1})$
Input:	$\texttt{ct}_{j,k} \in R^2_{n,Q_i}$, $j = 1, \ldots, 2k_2$, $k = 1, \ldots, k_1$ ciphertexts,
	$\texttt{ek}_i \in R^2_{n,Q_i}$ evaluation key at level i,
	$Q_i \in \mathbb{N}$ integer modulus,
	$Q_{i-1} \in \mathbb{N}$ integer modulus with $Q_i = q_i Q_{i-1}$.
Output:	$\texttt{ct} \in R^2_{n,Q_{i-1}}$.
Step 1.	For j from 1 to $2k_2$ do
	$\texttt{ct}_j := \texttt{Linearcombo}(\texttt{ct}_{j,1}, \ldots, \texttt{ct}_{j,k_1}, 1, \ldots, 1)$.
Step 2.	Initialize $\texttt{ct} := (0, 0, 0)$.
	For j from 1 to k_2 do
	$\texttt{ct} := \texttt{ct} + \texttt{Multiply}(\texttt{ct}_{2j-1}, \texttt{ct}_{2j})$.
Step 3.	Compute $\texttt{ct} := \texttt{Relinearize}(\texttt{ct}, \texttt{ek}_i)$.
Step 4.	Compute $\texttt{ct} := \texttt{Modreduce}(Q_i, Q_{i-1}, \texttt{ct})$.
Step 5.	Return $\texttt{ct}$.

Algorithm 17: Depth-1 Multiplication

$\texttt{ct}_{j,k}$ is a BFV ciphertext, Algorithm 17 will use `BFV.Multiply`, `BFV.Relinearize`, and `BFV.Modreduce`, while `Linearcombo` is identical for all three ciphertext types. Note that $\texttt{ek}_i \in R^2_{n,p_1Q_i}$ is assumed to match the ciphertext type of the $\texttt{ct}_{j,k}$'s. Our relinearization takes place after summing together our $\texttt{ct}_j$'s obtained from step 2, which are each a polynomial triple. This slightly improves our bounds, and is better from a computational perspective since we are only running `Relinearize` once in Algorithm 17.

To guarantee the amount of computation we can perform, we want to choose q_i so that the output of Algorithm 17 is always a ciphertext with noise bounded by ρ when all the $\texttt{ct}_{j,k}$ inputs have noise bounded by ρ. The precise bound on q_i is presented in Lemmas 4.1 and 4.2 for BFV and BGV, respectively.

Lemma 4.1 *For any $1 \leq i \leq \ell$, suppose $q_i > 9k_1k_2tn^2$ and $\delta_R \geq 16$. Then, for a collection of BFV ciphertexts at level i all with noise bounded by ρ, the output of Algorithm 17 is a BFV ciphertext at level $i-1$ with noise bounded by ρ.*

Lemma 4.2 *For any $1 \leq i \leq \ell$, suppose $q_i > 4k_1^2k_2tn^2$ and $\delta_R \geq 16$. Then, for a collection of BGV ciphertexts at level i all with noise bounded by ρ, the output of Algorithm 17 is a BGV ciphertext at level $i-1$ with noise bounded by ρ.*

For a similar depth-1 result in CKKS, we must use caution when finding conditions for q_i. The reason for this is that in CKKS, we do not have much flexibility to choose

q_i. For the standard scheme, it is always assumed that $q_i = \Delta$ for each $i \neq 0$. Thus, the best that we can do is to bound the error in general after computing the depth-1 algorithm. We can not force the error down within a constant after rescaling without the assumption that $q_i \gg \Delta$, which clearly contradicts the size of Δ needed in CKKS. We give the result on bounding CKKS noise below in Lemma 4.3.

Lemma 4.3 *Let i be such that $1 \leq i \leq \ell$, and suppose that $n^2 \leq \Delta$ and $q_i = \Delta$. Suppose we have a collection of CKKS ciphertexts at level i all with noise bounded by E. Furthermore, suppose that the corresponding messages $z_j \in \mathbb{C}^{n/2}$ each satisfy $\left\|z_j\right\|_\infty \leq Z$. Then Algorithm 17 results in a CKKS ciphertext at level $i-1$ with noise bounded by $2k_1k_2nEZ + \frac{k_1k_2En}{\Delta} + \frac{k_1^2k_2E^2n}{\Delta} + \frac{1}{8}$.*

One special case of depth 1 multiplication is the inner product of vectors. That is, given vectors of messages $\mathbf{m} = (m_1, \ldots, m_k) \in R_{n,t}^k$ and $\mathbf{m}' = (m_1', \ldots, m_k') \in R_{n,t}^k$, we want to compute $\langle \mathbf{m}, \mathbf{m}' \rangle \in R_{n,t}$ homomorphically. This can be thought of as one round of k products between the corresponding ciphertexts of $m_1, \ldots, m_k$ and $m_1', \ldots, m_k'$, followed by $k-1$ additions to sum them together. This is simply a depth-1 multiplication with $k_1 = 1$ and $k_2 = k$. Alternatively, a depth-1 multiplication with $k_1 = k$ and $k_2 = 1$ allows for k ciphertexts to be added together for two separate groups, followed by a single multiplication between the two sums. We argue that our proposed model allows for some more flexibility from a theoretical perspective, as we provide for additions both before and after multiplication at each modulus level.

Remark. Algorithm 17 also works for groups of ciphertext inputs of size less than k_1 with arbitrary linear combinations in step 1. That is, we can compute

$$\mathtt{ct}_j := \mathtt{Linearcombo}(\mathtt{ct}_{j,1}, \ldots, \mathtt{ct}_{j,k_j'}, \alpha_{j,1}, \ldots, \alpha_{j,k_j'})$$

so long as $k_j' \leq k_1$ for each j. Furthermore, if for each j we have

$$\sum_{\omega=1}^{k_j'} |\alpha_{j,\omega}| \leq k_1,$$

then Lemmas 4.1, 4.2, and 4.3 still apply.

4.2 Operations in the Residue Number System

Implementations of homomorphic encryption [27, 30, 31, 32] take advantage of the RNS variants of schemes [19, 21, 33, 34]. In our modified leveled homomorphic schemes, we would require that each q_i be chosen coprime to one another in order to use the Chinese remainder theorem. Additions and multiplications are computed componentwise (except for BFV multiplication), which provides the major computational advantage over computation modulo large integers. However, algorithms for

modulus reductions and relinearization need to be modified in order to avoid operations in large integers.

Basis Conversion in RNS. Suppose $q = q_0 \cdots q_{k-1}$ and $p = q_k \cdots q_{k+\ell-1}$ where $q_0, \ldots, q_{k-1}, q_k \ldots, q_{k+\ell-1}$ are distinct primes. Denote

$$\mathcal{B} = (q_0, \ldots, q_{k-1}), \quad \mathcal{C} = (q_k, \ldots, q_{k+\ell-1})$$

as two arbitrary ordered sets which we call *bases*. For an element $a \in R_{n,q}$, we denote $[a]_{\mathcal{B}} \in \prod_{j=0}^{k-1} R_{n,q_j}$ as the vector of CRT components of a in basis $\mathcal{B}$. That is,

$$[a]_{\mathcal{B}} = (a^{(0)}, a^{(1)}, \ldots, a^{(k-1)}) = (a^{(j)})_{0 \le j \le k-1}$$

where $a^{(j)} := a \mod q_j$ for $0 \le j < k$. We need to compute $a \bmod p$, that is, $[a]_{\mathcal{C}}$. To do this, let $\hat{q}_j = q/q_j \in \mathbb{Z}$ and $r_j = \hat{q}_j^{-1} a^{(j)} \bmod q_j$ for $0 \le j \le k-1$, where $\|r_j\|_\infty \le q_j/2$. Let

$$\tilde{a} = \sum_{j=0}^{k-1} \hat{q}_j r_j.$$

One can check that $\tilde{a} \equiv a_j \pmod{q_j}$ for $0 \le j \le k-1$, hence $\tilde{a} \equiv a \pmod{q}$. Then one can compute $\tilde{a} \bmod q_j$ for $k \le j \le k+\ell-1$ to get $[\tilde{a}]_{\mathcal{C}}$. This yields Algorithm 18 below from [21] and [34].

	$\texttt{Conv}([a]_{\mathcal{B}}, \mathcal{B}, \mathcal{C})$
Input:	$\mathcal{B} = (q_0, \ldots, q_{k-1})$ with $q = q_0 \cdots q_{k-1}$,
	$\mathcal{C} = (q_k, \ldots, q_{k+\ell-1})$ with $p = q_k \cdots q_{k+\ell-1}$,
	$[a]_{\mathcal{B}} = (a^{(0)}, \ldots, a^{(k-1)})$, RNS representation of $a \in R_{n,q}$ in basis $\mathcal{B}$.
Output:	$[\tilde{a}]_{\mathcal{C}} = (\tilde{a}^{(0)}, \ldots, \tilde{a}^{(\ell-1)})$, RNS representation of $\tilde{a} \in R_{n,p}$ in basis $\mathcal{C}$.
Step 1.	For $0 \le i \le k-1$, compute $r_i := [a^{(i)} \cdot \hat{q}_i^{-1}]_{q_i}$.
Step 2.	For $0 \le i \le \ell - 1$, compute $\tilde{a}^{(i)} := \left[\sum_{j=0}^{k-1} \hat{q}_j \cdot r_j \right]_{q_{k+i}}$.
Step 3.	Return $[\tilde{a}]_{\mathcal{C}} = (\tilde{a}^{(0)}, \ldots, \tilde{a}^{(\ell-1)})$.

Algorithm 18: Fast Basis Conversion

Lemma 4.4 ([21]) *Suppose the input of Algorithm 18 is the RNS representation in basis $\mathcal{B}$ of an element $a \in R_{n,q}$. Then, the output $[\tilde{a}]_{\mathcal{C}}$ is the RNS representation in basis $\mathcal{C}$ of an element $\tilde{a} \in R_{n,p}$ satisfying*

$$\tilde{a} = a + q \cdot e$$

for some $e \in R_n$ satisfying $|a + q \cdot e| \le q \cdot k/2$ and $|e| \le k/2$.

We note that the bound on e follows from the fact that $\|\tilde{a}\|_\infty \leq qk/2$. This means that $\tilde{a}$ is only an approximation of a. There are other fast basis conversions in the literature which give an exact switch (e.g., see [33]). For our purposes in the analysis of relinearization error however, the approximate switching in Algorithm 18 will suffice.

Modulus Reduction in RNS. Let p be a factor of Q, say $Q = qp$. For any polynomial $a \in R_{n,Q}$, we need to compute the rounding:

$$\left\lfloor \frac{q \cdot a}{Q} \right\rceil = \left\lfloor \frac{a}{p} \right\rceil \in R_{n,q}.$$

Suppose $q = q_0 \cdots q_{k-1}$ and $p = q_k \cdots q_{k+\ell-1}$ where $q_0, \ldots, q_k, \ldots, q_{k+\ell-1}$ are distinct primes. In RNS, a is represented as $(a^{(0)}, \ldots, a^{(k+\ell-1)})$ where

$$a \equiv a^{(i)} \pmod{q_i}, \qquad 0 \leq i \leq k+\ell-1. \tag{4}$$

By the Chinese remainder theorem, the solution a to equation (4) is unique modulo Q. Note that, for any two polynomials a and b with $a \equiv b \pmod{Q}$, we have

$$\left\lfloor \frac{q \cdot a}{Q} \right\rceil \equiv \left\lfloor \frac{q \cdot b}{Q} \right\rceil \pmod{q}.$$

This is true even if q is not a factor of Q. Hence, we can use any solution a to (4) in the rounding.

Define $\hat{Q}_i = Q/q_i$ and

$$r_i := \hat{Q}_i^{-1} a^{(i)} \bmod q_i, \qquad 0 \leq i \leq k+\ell-1,$$

where the coefficients of r_i are bounded by $q_i/2$. Then a solution of (4) is

$$a = \sum_{i=0}^{k+\ell-1} \hat{Q}_i \, r_i = p \sum_{i=0}^{k-1} \frac{q}{q_i} r_i + q \sum_{i=k}^{\ell-1} \frac{p}{q_i} r_i.$$

Note that

$$\frac{a}{p} = \sum_{i=0}^{k-1} \frac{q}{q_i} r_i + \sum_{i=k}^{\ell-1} \frac{q}{q_i} r_i.$$

The first sum has integer coefficients, and we only need to round the second sum. Let

$$w = \left\lfloor \sum_{i=k}^{\ell-1} \frac{q}{q_i} r_i \right\rceil.$$

Then

$$\frac{a}{p} = \sum_{i=0}^{k-1} \frac{q}{q_i} r_i + w + e \tag{5}$$

where $e \in \mathbb{R}[x]/(\phi(x))$ with $\|e\|_\infty \leq 1/2$. Also, note that, for $0 \leq j \leq k-1$,

$$\sum_{i=0}^{k-1} \frac{q}{q_i} r_i \equiv p^{-1} a^{(j)} \bmod q_j.$$

This gives us the algorithm 19 below that matches Algorithm 1.

	$\texttt{RNS.BFV.ModReduce}([a]_{\mathcal{D}}, \mathcal{D}, \mathcal{B})$
Input:	$\mathcal{D} = (q_0, \cdots, q_{k+\ell-1})$,
	$\mathcal{B} = (q_0, \ldots, q_{k-1})$ with $q = q_0 \cdots q_{k-1}$ and $p = q_k \cdots q_{k+\ell-1}$,
	$[a]_{\mathcal{D}} = (a^{(0)}, \ldots, a^{(k+\ell-1)})$, representation of $a \in R_{n,Q}$ in basis $\mathcal{D}$.
Output:	$[b]_{\mathcal{B}} = (b^{(0)}, \ldots, b^{(k-1)})$, RNS representation of $b \in R_{n,q}$ in basis $\mathcal{B}$.
Step 1.	For $k \leq i \leq k+\ell-1$ and $\hat{p}_i = p/q_i$, compute
	$r_i := [\hat{p}_i^{-1} \cdot a^{(i)}]_{q_i}$.
Step 2.	Compute
	$w := \left\lfloor \sum_{i=k}^{k+\ell-1} \frac{q}{q_i} \cdot r_i \right\rceil$ in R_n.
Step 3.	For $0 \leq j \leq k-1$, compute
	$b^{(j)} := [p^{-1} a^{(j)} + w]_{q_j}$.
Step 4.	Return $[b]_{\mathcal{B}} = (b^{(0)}, \ldots, b^{(k-1)}) \in \prod_{i=0}^{k-1} R_{n,q_i}$.

Algorithm 19: RNS BFV Modulus Reduction (v1)

The w above gives a rounding error at most $1/2$, however, it might be too expensive to compute, as its coefficients are too large. Next, we derive a faster rounding method, with a slightly larger rounding error. Let $\hat{p}_i = p/q_i$ and $v_i := \hat{p}_i^{-1} a^{(i)} \bmod q_i$ with $\|v_i\|_\infty \leq p/2$ for $k \leq i \leq k+\ell-1$. Then

$$v = \sum_{i=k}^{k+\ell-1} \hat{p}_i v_i$$

satisfies $v \equiv a \pmod{p}$ and $\|v\|_\infty \leq (p\ell)/2$. Let $u = \frac{a-v}{p}$, which has integer coefficients. Then

$$\frac{a}{p} = u + e \tag{6}$$

where $e = v/p \in \mathbb{R}[x]/(\phi(x))$ with $\|e\|_\infty \leq \ell/2$. In RNS, $u \bmod q$ can be obtained by first computing $v \bmod q_j$, $0 \leq j \leq k-1$, via basis conversion from p to q. Algorithm 20 describes the procedure, while Lemma 4.5 shows the noise bound. We exclude the proof of Lemma 4.5 from our appendix, as it is clear from the previous discussion.

	$\texttt{RNS.BFV.ModReduce}([a]_{\mathcal{D}}, \mathcal{D}, \mathcal{B})$
Input:	$\mathcal{D} = (q_0, \cdots, q_{k+\ell-1})$,
	$\mathcal{B} = (q_0, \ldots, q_{k-1})$ with $q = q_0 \cdots q_{k-1}$ and $p = q_k \cdots q_{k+\ell-1}$,
	$[a]_{\mathcal{D}} = (a^{(0)}, \ldots, a^{(k+\ell-1)})$, representation of $a \in R_{n,Q}$ in basis $\mathcal{D}$.
Output:	$[b]_{\mathcal{B}} = (b^{(0)}, \ldots, b^{(k-1)})$, RNS representation of $b \in R_{n,q}$ in basis $\mathcal{B}$.
Step 1.	Let $\mathcal{C} = \mathcal{D} \setminus \mathcal{B} = (q_k, \ldots, q_{k+\ell-1})$. Compute
	$(v^{(0)}, \ldots, v^{(k-1)}) := \texttt{Conv}((a^{(k)}, \ldots, a^{(k+\ell-1)}), \mathcal{C}, \mathcal{B})$.
Step 2.	For $0 \le j \le k-1$, compute
	$b^{(j)} := p^{-1} \cdot (a^{(j)} - v^{(j)}) \bmod q_j$.
Step 3.	Return $[b]_{\mathcal{B}} = (b^{(0)}, \ldots, b^{(k-1)}) \in \prod_{i=0}^{k-1} R_{n,q_i}$.

Algorithm 20: RNS BFV Modulus Reduction (v2)

Lemma 4.5 *Let the RNS representation of $a \in R_{n,Q}$ be the input of Algorithm 20 and the RNS representation of $b \in R_{n,q}$ the output. Then,*

$$\frac{a}{p} = b + e$$

for some $e \in \mathbb{R}[x]/(\phi(x))$ with $\|e\|_\infty \le \ell/2$.

Next, we show a BGV modulus reduction in RNS that matches Algorithm 2. When q is a factor of Q, say $Q = qp$, note that $(-a_0 q t^{-1}) \bmod Q$ is the same as $q(-a_0 t^{-1} \bmod p)$. Hence, Algorithm 2 can be simplified as follows:

$$\omega_a := [-a_0 t^{-1}]_p \quad \text{and} \quad \omega_b := [-b_0 t^{-1}]_p$$
$$a'_0 := \left[\frac{a_0 + t\,\omega_a}{p}\right]_q \quad \text{and} \quad b'_0 := \left[\frac{b_0 + t\,\omega_b}{p}\right]_q.$$

Algorithm 21 describes the above procedure for reduction of one polynomial, while Lemma 4.6 shows the noise bound. We exclude the proof of Lemma 4.6, since it is clear from the discussion and the proof of Lemma 2.3.

Lemma 4.6 *Let the RNS representation of $a \in R_{n,Q}$ be the input of Algorithm 21 and the RNS representation of $b \in R_{n,q}$ the output. Then,*

$$\frac{a}{p} = b + t \cdot e$$

for some $e \in \mathbb{R}[x]/(\phi(x))$ with $\|e\|_\infty \le \ell/2$.

	$\texttt{RNS.BGV.ModReduce}([a]_{\mathcal{D}}, \mathcal{D}, \mathcal{B})$
Input:	$\mathcal{D} = (q_0, q_1, \cdots, q_{k+\ell-1})$,
	$\mathcal{B} = (q_0, \ldots, q_{k-1})$ and $p = q_k \cdots q_{k+\ell-1}$,
	$[a]_{\mathcal{D}} = (a^{(0)}, \ldots, a^{(k+\ell-1)})$, representation of $a \in R_{n,Q}$ in basis $\mathcal{D}$.
Output:	$[b]_{\mathcal{B}} = (b^{(0)}, \ldots, b^{(k-1)})$, representation of $b \in R_{n,q}$ in basis $\mathcal{B}$.
Step 1.	For $k \leq i \leq k + \ell - 1$, compute
	$w^{(i)} := [-t^{-1}a^{(i)}]_{q_i}$.
Step 2.	Let $\mathcal{C} = \mathcal{D} \setminus \mathcal{B} = (q_k, \ldots, q_{k+\ell-1})$. Compute
	$(u^{(0)}, \ldots, u^{(k-1)}) := \texttt{Conv}((w^{(k)}, \ldots, w^{(k+\ell-1)}), \mathcal{C}, \mathcal{B})$.
Step 3.	For $0 \leq i \leq k - 1$, compute
	$b_i := [p^{-1}(a^{(i)} + tu^{(i)})]_{q_i}$.
Step 4.	Return $(b^{(0)}, \ldots, b^{(k-1)}) \in \prod_{i=0}^{k-1} R_{n,q_i}$.

Algorithm 21: RNS BGV Modulus Reduction

For the CKKS rescaling procedure in RNS, we can again use the same procedure as the fast BFV modulus reduction in Algorithm 20. Said otherwise,

$$\texttt{RNS.CKKS.ModReduce = RNS.BFV.ModReduce}.$$

Note here that we specifically use (v2) of the RNS BFV Modulus reduction for RNS CKKS. Likewise, we can also use 4.5 for RNS CKKS when discussing the noise bound after performing `RNS.CKKS.Modreduce`.

Relinearization in RNS. For the rest of this section, fix $Q_\ell = q_0 \cdots q_\ell$ and $P = p_0 \cdots p_{k-1}$ with $q_0 \cdots q_\ell, p_0 \cdots p_{k-1}$ all coprime. For $0 \leq i \leq \ell$, let $Q_i = q_0 \cdots q_i$ and fix the ordered bases

$$\begin{aligned}
\mathcal{B} &= (q_0, \ldots, q_i), \\
\mathcal{C} &= (p_0, \ldots, p_{k-1}), \\
\mathcal{D} &= \mathcal{B} \cup \mathcal{C} = (q_0, \ldots, q_i, p_0, \ldots, p_{k-1}).
\end{aligned}$$

The goal of our relinearization algorithms in RNS will again be to closely match the previously outlined relinearizations for the classic variants in Algorithms 8 and 13. We first introduce the evaluation key generations for the three schemes, followed by their relinearization procedures. We should note that we only discuss the evaluation key generation here. For a full discussion on all key generations (e.g., secret and public keys) in RNS, see [19, 21, 33, 34].

We begin with RNS BFV. The procedure for evaluation key generation is given in Algorithm 22. Observe that this evaluation key generation is the same as the evaluation key generation in Algorithm 3, only computed in RNS.

	$\texttt{RNS.BFV.ek.Keygen}(\texttt{sk}, (q_0, \ldots, q_\ell, p_0, \ldots, p_{k-1}))$
Input:	$\texttt{sk} = s \in R_{n,3}$ secret key,
	$(q_0, \ldots, q_\ell, p_0, \ldots, p_{k-1})$ full RNS basis.
Output:	$\texttt{ek} = (\tilde{\texttt{k}}_0^{(j)}, \tilde{\texttt{k}}_1^{(j)})_{0 \leq j \leq k+\ell} \in \prod_{j=0}^{\ell} R_{n,q_j}^2 \times \prod_{j=0}^{k-1} R_{n,p_j}^2$ evaluation key.
Step 1.	Sample $(\tilde{\texttt{k}}_0^{(0)}, \ldots, \tilde{\texttt{k}}_0^{(k+\ell)}) \leftarrow U(\prod_{j=0}^{\ell} R_{n,q_j}^2 \times \prod_{j=0}^{k-1} R_{n,p_j}^2)$ and $\tilde{e} \leftarrow \chi_\rho$.
Step 2.	For $0 \leq j \leq \ell$ compute $\tilde{\texttt{k}}_1^{(j)} := [-\tilde{\texttt{k}}_0^{(j)} s + [P]_{q_j} s^2 + \tilde{e}]_{\phi(x), q_j}$.
Step 3.	For $0 \leq j \leq k-1$ compute $\tilde{\texttt{k}}_1^{(\ell+1+j)} := [-\tilde{\texttt{k}}_0^{(\ell+1+j)} s + \tilde{e}]_{\phi(x), p_j}$.
Step 4.	Return $\texttt{ek} = (\tilde{\texttt{k}}_0^{(j)}, \tilde{\texttt{k}}_1^{(j)})_{0 \leq j \leq k+\ell}$.

Algorithm 22: RNS BFV Evaluation Key Generation

For the RNS BGV scheme, the evaluation key generation is again a similar approach to the original evaluation key generation from Algorithm 9. Algorithm 23 gives the procedure for RNS BGV.

	$\texttt{RNS.BFV.ek.Keygen}(\texttt{sk}, (q_0, \ldots, q_\ell, p_0, \ldots, p_{k-1}))$
Input:	$\texttt{sk} = s \in R_{n,3}$ secret key,
	$(q_0, \ldots, q_\ell, p_0, \ldots, p_{k-1})$ full RNS basis.
Output:	$\texttt{ek} = (\tilde{\texttt{k}}_0^{(j)}, \tilde{\texttt{k}}_1^{(j)})_{0 \leq j \leq k+\ell} \in \prod_{j=0}^{\ell} R_{n,q_j}^2 \times \prod_{j=0}^{k-1} R_{n,p_j}^2$ evaluation key.
Step 1.	Sample $(\tilde{\texttt{k}}_0^{(0)}, \ldots, \tilde{\texttt{k}}_0^{(k+\ell)}) \leftarrow U(\prod_{j=0}^{\ell} R_{n,q_j}^2 \times \prod_{j=0}^{k-1} R_{n,p_j}^2)$ and $\tilde{e} \leftarrow \chi_\rho$.
Step 2.	For $0 \leq j \leq \ell$ compute $\tilde{\texttt{k}}_1^{(j)} := [-\tilde{\texttt{k}}_0^{(j)} s + [P]_{q_j} s^2 + t\tilde{e}]_{\phi(x), q_j}$.
Step 3.	For $0 \leq j \leq k-1$ compute $\tilde{\texttt{k}}_1^{(\ell+1+j)} := [-\tilde{\texttt{k}}_0^{(\ell+1+j)} s + t\tilde{e}]_{\phi(x), p_j}$.
Step 4.	Return $\texttt{ek} = (\tilde{\texttt{k}}_0^{(j)}, \tilde{\texttt{k}}_1^{(j)})_{0 \leq j \leq k+\ell}$.

Algorithm 23: RNS BGV Evaluation Key Generation

For the RNS CKKS scheme, the evaluation key generation is exactly the same as

the evaluation key generation for RNS BFV:

$$\texttt{RNS.CKKS.ek.Keygen} = \texttt{RNS.BFV.ek.Keygen}.$$

We are now ready to discuss the full RNS realinearization procedures. In these algorithms, we assume that we have obtained a vector of components $(c_0^{(j)}, c_1^{(j)}, c_2^{(j)})_{0\leq j\leq i} \in \prod_{j=0}^{i} R_{n,q_j}^3$ which are the RNS representation of some $(c_0, c_1, c_2) \in R_{n,Q_i}^3$ that is obtained after initial RNS multiplication for BFV, BGV, or CKKS. The initial RNS multiplication operations for BGV and CKKS are simply computed componentwise modulo the q_j's. The procedure for BFV is more complicated. We refer the reader to [33, 34] for details for the details on the initial RNS BFV multiplication.

For all three schemes, the RNS linearization procedure is given in Algorithm 24. Here, `RNS.Modreduce` is the RNS modulus reduction procedure for the chosen scheme, and txek is the corresponding evaluation key for that scheme. For instance, if we choose to run RNS relinearization for BGV, we use `RNS.BGV.ModReduce` in Step 4 with the corresponding evaluation key `ek` for BGV. We introduce Lemmas 4.7, 4.8 and 4.9 to prove correctness of the algorithm and our noise bound for RNS variant of BFV, BGV and CKKS, respectively.

Lemma 4.7 *Let* `ct` *be the output of Algorithm 24 and suppose the input* $(c_0^{(j)}, c_1^{(j)}, c_2^{(j)})_{0\leq j\leq i}$ *is the RNS representation of some* $(c_0, c_1, c_2) \in R_{n,Q_i}^3$ *satisfying*

$$c_0 + c_1 s + c_2 s^2 \equiv D_{Q_i}[m_0 m_1]_{\phi(x),t} + e' \mod (\phi(x), Q_i)$$

for $\|e'\|_\infty \leq E$*. If* $P \geq 6Q_i$*,* $\delta_R \geq 16$*, and* $k > i$*, then* `ct` *is the RNS representation of a BFV ciphertext with noise bounded by* $E + \frac{1}{8}\delta_R^2 k$*.*

Lemma 4.8 *Let* `ct` *be the output of Algorithm 24 and suppose the input* $(c_0^{(j)}, c_1^{(j)}, c_2^{(j)})_{0\leq j\leq i}$ *is the RNS representation of some* $(c_0, c_1, c_2) \in R_{n,Q_i}^3$ *satisfying*

$$c_0 + c_1 s + c_2 s^2 \equiv [m_0 m_1]_{\phi(x),t} + te' \mod (\phi(x), Q_i)$$

for $\|e'\|_\infty \leq E$*. If* $P \geq 6Q_i$*,* $\delta_R \geq 16$*, and* $k > i$*, then* `ct` *is the RNS representation of a BGV ciphertext with noise bounded by* $E + \frac{1}{8}\delta_R^2 k$*.*

Lemma 4.9 *Let* `ct` *be the output of Algorithm 24 and suppose the input* $(c_0^{(j)}, c_1^{(j)}, c_2^{(j)})_{0\leq j\leq i}$ *is the RNS representation of some* $(c_0, c_1, c_2) \in R_{n,Q_i}^3$ *satisfying*

$$c_0 + c_1 s + c_2 s^2 \equiv m_0 m_1 + e' \mod (\phi(x), Q_i)$$

for $\|e'\|_\infty \leq E$*. If* $P \geq 6Q_i$*,* $\delta_R \geq 16$*, and* $k > i$*, then* `ct` *is the RNS representation of a CKKS ciphertext with noise bounded by* $E + \frac{1}{8}\delta_R^2 k$*.*

	$\texttt{RNS.Relinearize}([(c_0, c_1, c_2)]_{\mathcal{B}}, [\texttt{ek}]_{\mathcal{D}})$
Input:	$\mathcal{B} = (q_0, \ldots, q_i)$ and $\mathcal{D} = (q_0, \ldots, q_i, p_0, \ldots, p_{k-1})$ bases, $[(c_0, c_1, c_2)]_{\mathcal{B}} = (c_0^{(j)}, c_1^{(j)}, c_2^{(j)})_{0 \leq j \leq i} \in \prod_{j=0}^{i} R_{n,q_j}^3$, $[\texttt{ek}]_{\mathcal{D}} = (\tilde{\texttt{k}}_0^{(j)}, \tilde{\texttt{k}}_1^{(j)})_{0 \leq j \leq i+k} \in \prod_{j=0}^{i} R_{n,q_j}^2 \times \prod_{j=0}^{k-1} R_{n,p_j}^2$ evaluation key.
Output:	$[\texttt{ct}]_{\mathcal{B}} = (a^{(j)}, b^{(j)})_{0 \leq j \leq i}$ RNS ciphertext.
Step 1.	Compute $(\tilde{c}_2^{(0)}, \ldots, \tilde{c}_2^{(k-1)}) := \texttt{Conv}((c_2^{(0)}, \ldots, c_2^{(i)}), \mathcal{B}, \mathcal{C})$.
Step 2.	For $0 \leq j \leq i$ compute $\hat{a}^{(j)} := [c_2^{(j)} \tilde{\texttt{k}}_0^{(j)}]_{\phi(x), q_j}$, $\hat{b}^{(j)} := [c_2^{(j)} \tilde{\texttt{k}}_1^{(j)}]_{\phi(x), q_j}$.
Step 3.	For $0 \leq j \leq k-1$ compute $\hat{a}^{(i+1+j)} := [\tilde{c}_2^{(j)} \tilde{\texttt{k}}_0^{(i+1+j)}]_{\phi(x), p_j}$, $\hat{b}^{(i+1+j)} := [\tilde{c}_2^{(j)} \tilde{\texttt{k}}_1^{(i+1+j)}]_{\phi(x), p_j}$.
Step 4.	Compute $(\hat{c}_1^{(0)}, \ldots, \hat{c}_1^{(i)}) := \texttt{RNS.Modreduce}((\hat{a}_0^{(0)}, \ldots, \hat{a}_0^{(i+k)}), \mathcal{D}, \mathcal{B})$, $(\hat{c}_0^{(0)}, \ldots, \hat{c}_0^{(i)}) := \texttt{RNS.Modreduce}((\hat{b}_0^{(0)}, \ldots, \hat{b}_0^{(i+k)}), \mathcal{D}, \mathcal{B})$.
Step 5.	For $0 \leq j \leq i$ compute $a^{(j)} := [c_1^{(j)} + \hat{c}_1^{(j)}]_{q_j}$, $b^{(j)} := [c_0^{(j)} + \hat{c}_0^{(j)}]_{q_j}$.
Step 6.	Return $\texttt{ct} = (a^{(j)}, b^{(j)})_{0 \leq j \leq i}$.

Algorithm 24: RNS Relinearization

In addition to the relinearization technique we opt for, we should also mention that there exists versions of the alternate relinearization technique from section 3.1 in RNS [33, 34]. In the RNS version, the element c_2 is essentially expanded into a bit decomposition twice: an expansion in the q_j's first, and then another expansion in a fixed base B for each component corresponding to each q_j. Though this technique is certainly viable, we opt for the outlined technique due to the smaller size of $\texttt{ek}$. In practice, both relinearization techniques are used together in a method known as *hybrid key switching* [35]. In practice, the noise bound for hybrid key switching is quite similar to what we've outlined in Lemmas 4.7, 4.8, and 4.9. The bound for hybrid key switching ranges from about $E + \frac{3}{8}\delta_R^2 k$ to $E + \frac{10}{8}\delta_R^2 k$ with our approach, using the noise bound from [16] and practical estimates of d_{num} from [9]. We refer the reader to [16, 35] for more details on hybrid key switching.

5 Lattices, Security, and Attacks

The security of homomorphic encryption schemes is based on the LWE problem over finite fields, which can be reduced to lattice problems. In this section, we'll give an overview of these lattice problems as well as various attacks on LWE. As this paper is more focused on noise reduction in homomorphic encryption schemes, we only provide a brief overview of security and attacks. For a more in-depth discussion on security, we refer the reader to various sources such as [23, 36, 29]. Decision-RLWE can be shown to be as hard as many worst case lattice problems [37]. There is also a brief mention of security reductions from RLWE to LWE in [29]. We'll discuss attacks on classic LWE rather than RLWE, since RLWE problems can be easily converted into LWE problems. Furthermore, all the best attack algorithms are for LWE instead of RLWE.

5.1 Lattices and Lattice Problems

Let V be an $\mathbb{R}$-vector space with $\dim(V) = m$. Λ is called a *lattice* if Λ is a discrete additive subgroup of V and each point of Λ is isolated (that is, no points in Λ are arbitrarily close to each other). In general, a lattice can be generated in the following way: given a matrix $B = (b_1, \ldots, b_n) \in \mathbb{R}^{m \times n}$ with independent columns, a lattice can be defined via

$$\Lambda = \{y \in \mathbb{R}^m : y = Bx, x \in \mathbb{Z}^n\}.$$

Here, $\Lambda \subseteq \mathbb{R}^m$ is an n-dimensional lattice. We call B a lattice basis. For a lattice Λ defined by lattice basis B, the volume $\mathrm{vol}(\Lambda)$ is defined as

$$\mathrm{vol}(\Lambda) = \sqrt{\det(B^T B)}$$

which can be proved to be independent of the choice of basis. Let

$$\lambda(\Lambda) = \min\{\|x\|_2 : x \in \Lambda, x \neq 0\}.$$

Definitions 5.1, 5.2, and 5.3 describe a few instances of well studied lattice problems for a lattice Λ [29].

Definition 5.1 (SVP) *The shortest vector problem (SVP) is as follows: Given a basis B of Λ, find a vector $v \in \Lambda$ such that $\|v\|_2 = \lambda(\Lambda)$.*

Definition 5.2 (γ-SVP) *The γ-approximate shortest vector problem (γ-SVP) is as follows: Given a basis B of Λ, find a nonzero vector $v \in \Lambda$ such that $\|v\|_2 \leq \gamma \cdot \lambda(\Lambda)$.*

Definition 5.3 (γ-GapSVP) *The γ-gap shortest vector problem (γ-GapSVP) is as follows: Given a basis B of Λ and a real number $r > 0$, output "yes" if $\lambda(\Lambda) \leq r$ and output "no" if $\lambda(\Lambda) > \gamma \cdot r$.*

These lattice problems are examples of NP-hard problems in the worst case. Regev [23] provides a reduction from an instance of $\gamma-$GapSVP to Decision-LWE [36], meaning that LWE is at least as hard as γ-GapSVP. For potential attacks on LWE based schemes, we'll discuss attack strategies outlined in [23, 29].

q-ary Lattices. In lattice-based cryptography, a class of lattices that is particularly important is q-ary lattices. We say Λ is a q-ary lattice if Λ is a lattice such that $q\mathbb{Z}^m \subseteq \Lambda \subseteq \mathbb{Z}^m$. In particular, given a matrix $A \in \mathbb{Z}^{m\times n}$, the following are q-ary lattices

$$\Lambda_q(A) = \{x \in \mathbb{Z}^m : x \equiv Ay \mod q \text{ with } y \in \mathbb{Z}^n\},$$
$$\Lambda'_q(A) = \{x \in \mathbb{Z}^m : x^T A \equiv 0 \mod q\}.$$

It is also worth mentioning that although the matrix A defines both of these lattices, A is not necessarily a lattice basis for these lattices. To find a lattice basis of $\Lambda_q(A)$, we can perform column operations on A modulo q, permuting the rows if necessary, to obtain a matrix

$$\begin{pmatrix} I_n \\ A_1 \end{pmatrix} \in \mathbb{Z}_q^{m\times n}$$

where I_n is an $n \times n$ identity matrix and $A_1 \in \mathbb{Z}_q^{(m-n)\times n}$ with high probability. Let

$$B = \begin{pmatrix} I_n & 0 \\ A_1 & qI_{m-n} \end{pmatrix} \in \mathbb{Z}^{m\times m}. \tag{7}$$

Then B has rank m and is a lattice basis for $\Lambda_q(A)$. Since $\Lambda_q(A)$ is a full rank lattice, $\text{vol}(\Lambda_q(A)) = |\det(B)| = q^{m-n}$.

To get a lattice basis for $\Lambda'_q(A)$, let A_1 be as above. Note that the solution space for $x^T A = 0 \mod q$ is spanned by the columns of the matrix

$$\begin{pmatrix} -A_1^T \\ I_{m-n} \end{pmatrix} \in \mathbb{Z}^{m\times(m-n)}.$$

Then, a basis for $\Lambda'_q(A)$ is

$$B' = \begin{pmatrix} -A_1^T & qI_n \\ I_{m-n} & 0 \end{pmatrix} \in \mathbb{Z}^{m\times m}.$$

The volume of this lattice is $\text{vol}(\Lambda'_q(A)) = q^n$.

Gaussian Heuristic. By the Gaussian Heuristic, for a lattice Λ of rank m, we expect its shortest vector to be of length

$$\sqrt{\frac{m}{2\pi \cdot \exp(1)}}\text{vol}(\Lambda)^{1/m}$$

on average [23, 38], where $\exp(1) = 2.7182\ldots$ is the exponential function evaluated at 1. As the lattice $\Lambda_q(A)$ has volume q^{m-n} and A_1 is uniform random in $\mathbb{Z}_q^{(m-n)\times n}$, we can use the Gaussian Heuristic and expect the shortest vector in $\Lambda_q(A)$ to be of length

$$\sqrt{\frac{m}{2\pi \cdot \exp(1)}} q^{1-n/m} \tag{8}$$

on average. Similarly for $\Lambda_q'(A)$, we expect its shortest vector to be of length

$$\sqrt{\frac{m}{2\pi \cdot \exp(1)}} q^{n/m}$$

on average.

5.2 LWE Attack Strategies

First, recall the LWE problems outlined in Section 2.2. Fix $s \in \mathbb{Z}_q^n$ which is secret. We sample $e_i \leftarrow \chi(\mathbb{Z})$ from some desired distribution χ such that $\|e_i\|_\infty \leq \rho$, where ρ is a desired parameter. Then, we sample a uniform random $a_i \in \mathbb{Z}_q^n$ and calculate b_i via $b_i = [-\langle a_i, s\rangle + e_i]_q$. The ordered pair $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ is called an *LWE sample*. The Search-LWE problem is to find s given many LWE samples. The Decision-LWE problem is given many samples that are either LWE samples or sampled uniform randomly, decide which distributions the samples are drawn from [25]. If we sample m times, we can instead think of the LWE samples as the matrix equation

$$b \equiv As + e \mod q \tag{9}$$

where $b \in \mathbb{Z}_q^m$ is the vector of b_i's, $A \in \mathbb{Z}_q^{m\times n}$ is the matrix of a_i's, and $e \in \mathbb{Z}_q^m$ is the vector of e_i's.

Dual Attacks via SVP. To solve Decision-LWE, we employ a dual attack. Let A be the matrix defined in equation 9. In dual attacks, we wish to find a short vector $v \in \Lambda_q'(A)$ since $\langle v, b\rangle = \langle v, As + e\rangle = \langle v, e\rangle \mod q$. Since e is short, $\langle v, b\rangle$ is small. If an adversary can find a short vector $v \in \Lambda_q'(A)$, then the adversary can solve the Decision-LWE problem with a fair amount of confidence, since $\langle v, b\rangle$ would likely not be small for true random b. Thus, the attacker can distinguish LWE samples from true random samples with advantage. We refer the reader to [8, 23, 39, 40] for more details on dual attacks and the exact advantages based on sizes of e and v.

Primal Attacks via SVP. A common attack strategy for Search-LWE is with SVP. Let A be the matrix defined in equation 9 and B be computed from A as in equation 7. Let

$$\tilde{B} = \begin{pmatrix} B & b \\ 0 & 1 \end{pmatrix} \in \mathbb{Z}^{(m+1)\times(m+1)}$$

and let $\tilde{\Lambda}$ be the lattice defined by $\tilde{B}$, which has volume $\mathrm{vol}(\tilde{\Lambda}) = q^{m-n}$. Then equation 9 means that the vector $\binom{e}{1}$ is in $\tilde{\Lambda}$. By the Gaussian Heuristic in equation

8, we expect the shortest vector in the lattice generated by B to be of length about

$$\sqrt{\frac{m}{2\pi \cdot \exp(1)}} q^{1-n/m}$$

since the entries of A_1 are uniform random in $\mathbb{Z}_q$. In general, $\|e\|_2$ is smaller than this. Thus, the shortest vector in $\tilde{\Lambda}$ is likely to be $\binom{e}{1}$ with a significant probability. Thus, when e is small, we can solve SVP for the lattice $\tilde{\Lambda}$ to find e, and in turn solve LWE. The error distribution χ is crucial in determining the expected size of e, which we discuss thoroughly in the next subsection. We refer the reader to [40, 41, 29] for more details on primal attacks.

Lattice Basis Reduction Algorithms. Several algorithms employ these strategies, as well as others, in order to solve LWE. Algorithms in practice for solving lattice problems include algorithms such as LLL [42], BKW [43], and BKZ [44], which are discussed thoroughly in [8] and [23]. We'll primarily discuss BKZ, as it seems to currently be the best algorithm for lattice reduction. The basic idea behind BKZ is to solve SVP for sublattices of dimension k, which is known as the *block size* in BKZ.

Let $B = (b_0, \ldots, b_{m-1})$ be a lattice basis for Λ, ordered so that b_0 is the shortest vector in B. Then, there is a constant γ_0 so that

$$\|b_0\|_2 = \gamma_0^m \mathrm{vol}(\Lambda)^{1/m}.$$

We call γ_0^m the Hermite factor and γ_0 the root-Hermite factor of the lattice basis B. The Hermite factor is crucial in determining cost and runtime of lattice reduction algorithms, especially BKZ. For a block size k in BKZ, Chen [45] shows that it is expected that the algorithm output a lattice basis B with γ_0 satisfying

$$\lim_{m\to\infty} \gamma_0 \approx \left(\frac{k}{2\pi \cdot \exp}(\pi k)^{\frac{1}{k}}\right)^{\frac{1}{2(k-1)}}. \tag{10}$$

In practice, the limiting factor from equation 10 is used to estimate γ_0 for a finite dimensional lattice [23]. One can compute a lattice basis B with root-Hermite factor γ_0 with the estimate in equation 10. For ease of analysis, γ_0 is approximated by either $\gamma_0 = k^{\frac{1}{2k}}$ or $\gamma_0 = 2^{\frac{1}{k}}$. Albrecht et al. [23] show that for block sizes $50 \leq k \leq 250$, $2^{\frac{1}{k}}$ actually approximates the estimate of γ_0 from equation 10 better than $k^{\frac{1}{2k}}$.

To determine γ_0, we look at three different cases dependent on the error distribution χ. If $\chi = \mathcal{D}_{\mathbb{Z},\alpha q}$ in our LWE instance, we expect $\|e\|_2 \approx \alpha q\sqrt{m}$ [2, 46]. To prove many theoretical security results, α and q are chosen so that $\alpha q = \sqrt{n}$ [23], meaning we can expect $\left\|\binom{e}{1}\right\|_2 \approx \sqrt{nm}$. Then, we can solve for γ_0 via

$$\sqrt{nm} = \gamma_0^{m+1} q^{\frac{m-n}{m+1}},$$

or equivalently,

$$\log(\gamma_0) = \frac{\log(\sqrt{nm}) - \log(q^{\frac{m-n}{m+1}})}{m+1}$$

as the lattice generated by $\tilde{B}$ is an $(m+1)$-dimensional lattice with volume q^{m-n}. Depending on the value of m, we can choose q so that $\binom{e}{1}$ is likely to be the shortest vector in $\tilde{\Lambda}$ based on the estimate from the Gaussian Heuristic in equation 8. For instance, if $m = 3n/2$, we could choose $q > n^{4.5}$ since

$$\sqrt{\frac{m}{2\pi \cdot \exp(1)}} q^{1-n/m} = \sqrt{\frac{m}{2\pi \cdot \exp(1)}} q^{1/3} = \sqrt{\frac{m}{2\pi \cdot \exp(1)}} n\sqrt{n} \gg \sqrt{nm} \approx \|e\|_2 .$$

For practical purposes, the homomorphic encryption standard [8] and several implementations [9, 26] use $\chi = \mathcal{D}_{\mathbb{Z},\alpha q}$ with $\alpha q = 8$, resulting in a standard deviation of $\sigma \approx 3.2$. This means we can expect $\left\| \binom{e}{1} \right\|_2 \approx 8\sqrt{m}$, which we can again use to solve for the root-Hermite factor γ_0 with

$$\log(\gamma_0) = \frac{\log(\sqrt{64nm}) - \log(q^{\frac{m-n}{m+1}})}{m+1}.$$

For $m = 3n/2$, choosing $q > n^3$ suffices since for large n, as

$$\sqrt{\frac{m}{2\pi \cdot \exp(1)}} q^{1-n/m} = \sqrt{\frac{m}{2\pi \cdot \exp(1)}} n \gg 8\sqrt{m} \approx \|e\|_2 .$$

In our proposed schemes, χ is a discrete uniform distribution in $[-n, n]$ with standard deviation

$$\sigma = \sqrt{\frac{(2n+1)^2 - 1}{12}} = \sqrt{\frac{4n^2 + 4n}{12}} \approx \frac{n}{\sqrt{3}}.$$

We can then expect $\left\| \binom{e}{1} \right\|_2 \approx \sqrt{n^2 m/3}$ as a rough estimate, allowing us to choose γ_0 accordingly by solving

$$\log(\gamma_0) = \frac{\log(\sqrt{n^2 m/3}) - \log(q^{\frac{m-n}{m+1}})}{m+1}.$$

If $m = 3n/2$, a modulus size of $q > n^6$ is sufficient to ensure $\binom{e}{1}$ is the shortest vector of $\tilde{\Lambda}$ in our case of a discrete uniform error distribution for χ.

After decided on the block size k, one can estimate the cost of BKZ as follows. For algorithms to solve SVP for lattices of rank k, the fastest known classical algorithm (using sieving) [47] runs in time $2^{0.292k+o(k)}$. The fastest known quantum algorithm [48] runs in time $2^{0.265k+o(k)}$. With BKZ, we can have up to $8m$ calls to an oracle solving SVP using a sieving algorithm, meaning that the total costs for BKZ we can expect are $8m \cdot 2^{0.292k+o(k)}$ classically and $8m \cdot 2^{0.265k+o(k)}$ quantumly [8].

In summary, to estimate the total cost of attack, we first determine the expected size of e based on the error distribution χ. After finding this expected size of e, we

can calculate the root-Hermite factor γ_0, which then allows us to determine the block size k and the total cost of BKZ. Though a thorough security analysis is still to be conducted on our strategy for picking leveled homomorphic encryption parameters, extensive research and estimates have been performed on the best known algorithms for solving LWE as briefly shown above. We point the reader to [8, 9, 23, 49] for a more complete discussion and further references on security.

6 Conclusions

In this paper, we have presented a detailed mathematical foundation for the BGV, BFV, and CKKS homomorphic encryption schemes, aligning our work with the functionalities proposed in recent homomorphic encryption standards. By providing protocol algorithms and correctness proofs, we have ensured that these schemes are not only theoretically sound but also practical for implementation. Our proposed improvements, particularly in noise management and leveled homomorphic computation, enhance the efficiency and applicability of these schemes by reducing ciphertext expansion and storage requirements. In future works, we plan to extend these theoretical results to RNS variants of schemes, as well as analyze implications of noise bounds and control of accuracy precision for homomorphic computation in CKKS.

References

[1] Gentry C. A fully homomorphic encryption scheme (PhD thesis); 2009. Stanford University. Available from: https://crypto.stanford.edu/craig/craig-thesis.pdf.

[2] Regev O. On lattices, learning with errors, random linear codes, and cryptography. J ACM. 2009 sep;56(6). Available from: https://doi.org/10.1145/1568318.1568324.

[3] Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings. In: Gilbert H, editor. Advances in Cryptology – EUROCRYPT 2010. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010. p. 1-23.

[4] Brakerski Z, Gentry C, Vaikuntanathan V. Fully homomorphic encryption without bootstrapping; 2011. Cryptology ePrint Archive, Report 2011/277. Available from: https://ia.cr/2011/277.

[5] Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans Comput Theory. 2014 jul;6(3). Available from: https://doi.org/10.1145/2633600.

[6] Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption; 2012. https://ia.cr/2012/144. Cryptology ePrint Archive, Report 2012/144.

[7] Cheon JH, Kim A, Kim M, Song Y. Homomorphic encryption for arithmetic of approximate numbers. In: Takagi T, Peyrin T, editors. Advances in Cryptology – ASIACRYPT 2017. Cham: Springer International Publishing; 2017. p. 409-37.

[8] Albrecht M, Chase M, Chen H, Ding J, Goldwasser S, Gorbunov S, et al.. Homomorphic encryption standard; 2019. Cryptology ePrint Archive, Paper 2019/939 (preprint). Available from: https://eprint.iacr.org/2019/939.

[9] Bossuat JP, Cammarota R, Cheon JH, Chillotti I, Curtis BR, Dai W, et al.. Security guidelines for implementing homomorphic encryption; 2024. Cryptology ePrint Archive, Paper 2024/463 (preprint). Available from: https://eprint.iacr.org/2024/463.

[10] Chillotti I, Gama N, Georgieva M, Izabachène M. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. In: Cheon JH, Takagi T, editors. Advances in Cryptology – ASIACRYPT 2016. Berlin, Heidelberg: Springer Berlin Heidelberg; 2016. p. 3-33.

[11] Chillotti I, Gama N, Georgieva M, Izabachène M. Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE. In: Takagi T, Peyrin

T, editors. Advances in Cryptology – ASIACRYPT 2017. Cham: Springer International Publishing; 2017. p. 377-408.

[12] Gao S. Efficient fully homomorphic encryption scheme; 2018. Cryptology ePrint Archive, Paper 2018/637 (preprint). Available from: https://eprint.iacr.org/2018/637.

[13] Case BM, Gao S, Hu G, Xu Q. Fully homomorphic encryption with k-bit arithmetic operations; 2019. Cryptology ePrint Archive, Paper 2019/521 (preprint). Available from: https://eprint.iacr.org/2019/521.

[14] Costache A, Smart NP. Which ring based somewhat homomorphic encryption scheme is best? In: Proceedings of the RSA Conference on Topics in Cryptology - CT-RSA 2016 - Volume 9610. Berlin, Heidelberg: Springer-Verlag; 2016. p. 325–340. Available from: https://doi.org/10.1007/978-3-319-29485-8_19.

[15] Bos JW, Lauter K, Loftus J, Naehrig M. Improved security for a ring-based fully homomorphic encryption scheme. In: Stam M, editor. Cryptography and Coding. Berlin, Heidelberg: Springer Berlin Heidelberg; 2013. p. 45-64.

[16] Kim A, Polyakov Y, Zucca V. Revisiting homomorphic encryption schemes for finite fields. In: Tibouchi M, Wang H, editors. Advances in Cryptology – ASIACRYPT 2021. Cham: Springer International Publishing; 2021. p. 608-39.

[17] Costache A, Curtis BR, Hales E, Murphy S, Ogilvie T, Player R. On the precision loss in approximate homomorphic encryption. In: Selected Areas in Cryptography – SAC 2023: 30th International Conference, Fredericton, Canada, August 14–18, 2023, Revised Selected Papers. Berlin, Heidelberg: Springer-Verlag; 2024. p. 325–345. Available from: https://doi.org/10.1007/978-3-031-53368-6_16.

[18] Costache A, Laine K, Player R. Evaluating the effectiveness of heuristic worst-case noise analysis in FHE. In: Computer Security – ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part II. Berlin, Heidelberg: Springer-Verlag; 2020. p. 546–565. Available from: https://doi.org/10.1007/978-3-030-59013-0_27.

[19] Gentry C, Halevi S, Smart NP. Homomorphic evaluation of the AES circuit. In: Safavi-Naini R, Canetti R, editors. Advances in Cryptology – CRYPTO 2012. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012. p. 850-67.

[20] Costache A, Nürnberger L, Player R. Optimisations and tradeoffs for HElib. In: Rosulek M, editor. Topics in Cryptology – CT-RSA 2023. Cham: Springer International Publishing; 2023. p. 29-53.

[21] Cheon JH, Han K, Kim A, Kim M, Song Y. A full RNS variant of approximate homomorphic encryption. Selected areas in cryptography : annual international workshop, SAC proceedings SAC. 2018;11349:347-68. Available from: https://api.semanticscholar.org/CorpusID:52977564.

[22] Lee E, Lee JW, Kim YS, No JS. Optimization of homomorphic comparison algorithm on RNS-CKKS scheme. IEEE Access. 2022;10:26163-76.

[23] Albrecht MR, Player R, Scott S. On the concrete hardness of learning with errors. Journal of Mathematical Cryptology. 2015;9(3):169-203. Available from: https://doi.org/10.1515/jmc-2015-0016.

[24] Lyubashevsky V, Peikert C, Regev O. A toolkit for ring-LWE cryptography. In: Johansson T, Nguyen PQ, editors. Advances in Cryptology – EUROCRYPT 2013. Berlin, Heidelberg: Springer Berlin Heidelberg; 2013. p. 35-54.

[25] Case B. Homomorphic encryption and cryptanalysis of lattice cryptography (PhD thesis); 2020. Clemson University. Available from: https://tigerprints.clemson.edu/all_dissertations/2635.

[26] Albrecht MR. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In: Coron JS, Nielsen JB, editors. Advances in Cryptology – EUROCRYPT 2017. Cham: Springer International Publishing; 2017. p. 103-29.

[27] Microsoft SEAL (release 4.1); 2023. Microsoft Research, Redmond, WA. https://github.com/Microsoft/SEAL.

[28] Yates K. Efficiency of homomorphic encryption schemes (MS thesis); 2022. Clemson University. Available from: https://tigerprints.clemson.edu/all_theses/3868.

[29] Player R. Parameter selection in lattice-based cryptography (PhD thesis); 2018. Royal Holloway, University of London. Available from: https://pure.royalholloway.ac.uk/ws/portalfiles/portal/29983580/2018playerrphd.pdf.

[30] Al Badawi A, Bates J, Bergamaschi F, Cousins DB, Erabelli S, Genise N, et al. OpenFHE: open-source fully homomorphic encryption library. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. WAHC'22. New York, NY, USA: Association for Computing Machinery; 2022. p. 53–63. Available from: https://doi.org/10.1145/3560827.3563379.

[31] HElib homomorphic encryption library; 2013. Available from: https://github.com/homenc/HElib.

[32] Halevi S, Shoup V. Design and implementation of HElib: a homomorphic encryption library; 2020. Cryptology ePrint Archive, Paper 2020/1481 (preprint). Available from: https://eprint.iacr.org/2020/1481.

[33] Halevi S, Polyakov Y, Shoup V. An improved RNS variant of the BFV homomorphic encryption scheme. In: Matsui M, editor. Topics in Cryptology – CT-RSA 2019. Cham: Springer International Publishing; 2019. p. 83-105.

[34] Bajard JC, Eynard J, Hasan MA, Zucca V. A Full RNS Variant of FV Like Somewhat Homomorphic Encryption Schemes. In: Avanzi R, Heys H, editors. Selected Areas in Cryptography – SAC 2016. Cham: Springer International Publishing; 2017. p. 423-42.

[35] Han K, Ki D. Better Bootstrapping for Approximate Homomorphic Encryption. In: Jarecki S, editor. Topics in Cryptology – CT-RSA 2020. Cham: Springer International Publishing; 2020. p. 364-90.

[36] Peikert C. A decade of lattice cryptography. Found Trends Theor Comput Sci. 2016 mar;10(4):283–424. Available from: https://doi.org/10.1561/0400000074.

[37] Peikert C, Regev O, Stephens-Davidowitz N. Pseudorandomness of ring-LWE for any ring and modulus. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing. STOC 2017. New York, NY, USA: Association for Computing Machinery; 2017. p. 461–473. Available from: https://doi.org/10.1145/3055399.3055489.

[38] Ducas L. Shortest vector from lattice sieving: a few dimensions for free; 2017. Cryptology ePrint Archive, Paper 2017/999 (preprint). Available from: https://eprint.iacr.org/2017/999.

[39] Ajtai M. Generating hard instances of lattice problems (extended abstract). In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. STOC '96. New York, NY, USA: Association for Computing Machinery; 1996. p. 99–108. Available from: https://doi.org/10.1145/237814.237838.

[40] Lindner R, Peikert C. Better key sizes (and attacks) for LWE-based encryption. In: Kiayias A, editor. Topics in Cryptology – CT-RSA 2011. Berlin, Heidelberg: Springer Berlin Heidelberg; 2011. p. 319-39.

[41] Lyubashevsky V, Micciancio D. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In: Halevi S, editor. Advances in Cryptology - CRYPTO 2009. Berlin, Heidelberg: Springer Berlin Heidelberg; 2009. p. 577-94.

[42] Lenstra AK, Lenstra HW, Lovász LM. Factoring polynomials with rational coefficients. Mathematische Annalen. 1982;261:515-34. Available from: https://api.semanticscholar.org/CorpusID:5701340.

[43] Blum A, Kalai A, Wasserman H. Noise-tolerant learning, the parity problem, and the statistical query model. Journal of the ACM. 2000 05;50:435-40.

[44] Schnorr C, Euchner M. Lattice basis reduction: improved practical algorithms and solving subset sum problems. Mathematical Programming. 1994 08;66:181-99.

[45] Chen Y. Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe (PhD thesis); 2013. Available from: https://api.semanticscholar.org/CorpusID:170791320.

[46] Regev O. The learning with errors problem (invited survey). In: 2010 IEEE 25th Annual Conference on Computational Complexity; 2010. p. 191-204.

[47] Becker A, Ducas L, Gama N, Laarhoven T. New directions in nearest neighbor searching with applications to lattice sieving. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '16. USA: Society for Industrial and Applied Mathematics; 2016. p. 10–24.

[48] Laarhoven T. Search problems in cryptography: from fingerprinting to lattice sieving (PhD thesis); 2016. Eindhoven University of Technology.

[49] Lepoint T, Naehrig M. A comparison of the homomorphic encryption schemes FV and YASHE. In: Pointcheval D, Vergnaud D, editors. Progress in Cryptology – AFRICACRYPT 2014. Cham: Springer International Publishing; 2014. p. 318-35.

Appendix A Proofs of Lemmas

Proof of Lemma 2.1. By assumption, $b_0 \equiv -a_0 s + e_0 \mod (\phi(x), Q)$. Therefore, there exists $r \in R_n$ such that

$$b \equiv -a_0 s + e_0 + Qr \mod \phi(x).$$

Since $a_0' = \lfloor \frac{qa_0}{Q} \rceil$ and $b_0' = \lfloor \frac{qb_0}{Q} \rceil$, there are polynomials $\epsilon_1, \epsilon_2 \in \mathbb{R}[x]/(\phi(x))$ such that $a_0' = \frac{qa_0}{Q} - \epsilon_1$ and $b_0' = \frac{qb_0}{Q} - \epsilon_2$ with $\|\epsilon_1\|_\infty, \|\epsilon_2\|_\infty \leq 1/2$. Then,

$$\begin{aligned} b_0' &= \frac{q}{Q} b_0 - \epsilon_2 \\ &\equiv -\frac{q}{Q} a_0 s + \frac{q}{Q} e_0 + qr - \epsilon_2 \mod \phi(x) \\ &\equiv -a_0' s + \frac{q}{Q} e_0 + qr - \epsilon_2 - \epsilon_1 s \mod \phi(x). \end{aligned}$$

Let $e_0' = \frac{q}{Q} e_0 - \epsilon_2 - \epsilon_1 s$. Then, $b_0' \equiv -a_0' s + e_0' \mod (\phi(x), q)$. Note that $\left\|e_0'\right\|_\infty \leq \frac{q}{Q} E + \frac{\delta_R \|s\|_\infty + 1}{2}$. By assumption $Q/q > \frac{2E}{\delta_R \|s\|_\infty - 1}$, so $\left\|e_0'\right\|_\infty < \delta_R \|s\|_\infty$.

Proof of Lemma 2.2. By assumption, we first note that $(a_0, b_0) \in R_{n,Q}^2$ satisfies $b_0 \equiv -a_0 s + D_Q m_0 + e_0 \mod (\phi(x), Q)$. Therefore, there is some $r_Q \in R_n$ such that $b_0 + a_0 s \equiv D_Q m_0 + e_0 + Q r_Q \mod \phi(x)$. Let $\epsilon_Q = Q/t - D_Q$, $\epsilon_q = q/t - D_q$, $\epsilon_1 = qa_0/Q - a_0'$, and $\epsilon_2 = qb_0/Q - b_0'$. Then,

$$b_0' = \frac{qb_0}{Q} - \epsilon_2 \equiv -\frac{qa_0 s}{Q} + \frac{qD_Q}{Q} m_0 + \frac{qe_0}{Q} - \epsilon_2 + qr_Q \mod \phi(x).$$

Note that as $D_Q = Q/t - \epsilon_Q$, we have that $qD_Q/Q = q/t - q\epsilon_Q/Q$. Since $q/t = D_q + \epsilon_q$, we have $qD_Q/Q = D_q + \epsilon_q - q\epsilon_Q/Q$. Therefore,

$$\begin{aligned} b_0' &\equiv -\frac{qa_0 s}{Q} + \frac{qD_Q}{Q} m_0 + \frac{qe_0}{Q} - \epsilon_2 + qr_Q \mod \phi(x) \\ &\equiv -a_0' s - \epsilon_1 s + D_q m_0 + (\epsilon_q - \frac{q\epsilon_Q}{Q}) m_0 + \frac{qe_0}{Q} - \epsilon_2 + qr_Q \mod \phi(x). \end{aligned}$$

Let $e_0' = \frac{qe_0}{Q} + (\epsilon_q - \frac{q\epsilon_Q}{Q}) m_0 - \epsilon_2 - \epsilon_1 s$. Then, $b_0' + a_0' s \equiv D_q m_0 + e_0' \mod (\phi(x), q)$. Furthermore if $Q > q$, $t|(Q-1)$, and $t|(q-1)$, then $D_Q = (Q-1)/t$ and $\epsilon_Q = 1/t$. Similarly, $D_q = (q-1)/t$ and $\epsilon_q = 1/t$. Then, $|\epsilon_q - q\epsilon_Q/Q| = \frac{1}{t}(1 - \frac{q}{Q}) < \frac{1}{t}$. So,

$$\begin{aligned} \left\|e_0'\right\|_\infty &= \left\| \frac{qe_0}{Q} + (\epsilon_q - q\epsilon_Q/Q) m_0 - \epsilon_2 - \epsilon_1 s \right\|_\infty \\ &\leq \frac{q}{Q} \|e_0\|_\infty + |\epsilon_q - q\epsilon_Q/Q| \frac{t}{2} + \|\epsilon_2\|_\infty + \|\epsilon_1 s\|_\infty \\ &\leq \frac{q}{Q} E + 1 + \frac{\delta_R \|s\|_\infty}{2}. \end{aligned}$$

By assumption $Q/q > \frac{2E}{\delta_R \|s\|_\infty - 2}$, so $\left\|e'_0\right\|_\infty < \delta_R \|s\|_\infty$.

Proof of Lemma 2.3. First, note that a'_0 and b'_0 are polynomials with integer coefficients since both $qa_0 + t\omega_a$ and $qb_0 + t\omega_b$ are equivalent to 0 modulo Q. Then, we have

$$\begin{aligned}
b'_0 + a'_0 s &\equiv \frac{qb_0 + t\omega_b}{Q} + \frac{qa_0 + t\omega_a}{Q} s \mod \phi(x) && (11)\\
&\equiv \frac{q}{Q}(b_0 + a_0 s) + \frac{t}{Q}(\omega_b + \omega_a s) \mod \phi(x) && (12)\\
&\equiv \frac{q}{Q}(m_0 + te_0) + \frac{t}{Q}(\omega_b + \omega_a s) + qr \mod \phi(x) && (13)\\
&\equiv m_0 + t\left(\frac{q-Q}{Qt} m_0 + \frac{q}{Q} e_0 + \frac{1}{Q}(\omega_b + \omega_a s)\right) + qr \mod \phi(x) && (14)\\
&\equiv m_0 + te'_0 \mod (\phi(x), q) && (15)
\end{aligned}$$

where $e'_0 = \frac{q-Q}{Qt} m_0 + \frac{q}{Q} e_0 + \frac{1}{Q}(\omega_b + \omega_a s)$. Since $Q > |q - Q|$, we have $|\frac{q-Q}{Qt}| < \frac{1}{t}$. Thus, we have that

$$\begin{aligned}
\left\|e'_0\right\|_\infty &= \left\| \frac{q-Q}{Qt} m_0 + \frac{q}{Q} e_0 + \frac{1}{Q}(\omega_b + \omega_a s) \right\|_\infty \\
&\le \left| \frac{q-Q}{Qt} \right| \|m_0\|_\infty + \frac{q}{Q} \|e_0\|_\infty + \frac{1}{Q} \|\omega_b\|_\infty + \frac{1}{Q} \|\omega_a s\|_\infty \\
&\le \frac{1}{2} + \frac{q}{Q} E + \frac{1}{2} + \frac{\delta_R \|s\|_\infty}{2} \\
&= \frac{q}{Q} E + 1 + \frac{\delta_R \|s\|_\infty}{2}.
\end{aligned}$$

By assumption $Q/q > \frac{2E}{\delta_R \|s\|_\infty - 2}$, so $\left\|e'_0\right\|_\infty < \delta_R \|s\|_\infty$.

Proof of Lemma 3.1. By assumption, the public key $\mathtt{pk} = (\mathtt{k}_0, \mathtt{k}_1)$ satisfies

$$\mathtt{k}_1 + \mathtt{k}_0 s \equiv e \mod (\phi(x), p_0 q)$$

for some noise $e \in R_n$ with $\|e\|_\infty \le \rho$. Then,

$$\begin{aligned}
b_0 + a_0 s &\equiv \mathtt{k}_1 u + e_2 + (\mathtt{k}_0 u + e_1) s \mod (\phi(x), p_0 q) \\
&\equiv -(\mathtt{k}_0 s + e) u + e_2 + (\mathtt{k}_0 u + e_1) s \mod (\phi(x), p_0 q) \\
&\equiv -\mathtt{k}_0 s u - eu + e_2 + \mathtt{k}_0 s u + e_1 s \mod (\phi(x), p_0 q) \\
&\equiv -eu + e_2 + e_1 s \mod (\phi(x), p_0 q)
\end{aligned}$$

Let $e_0 = -eu + e_2 + e_1 s$. Then, $b_0 + a_0 s \equiv e_0 \mod (\phi(x), p_0 q)$. Note that

$$\begin{aligned}\|e_0\|_\infty &= \|-eu + e_2 + e_1 s\|_\infty \\ &\leq \delta_R \|e\|_\infty \|u\|_\infty + \|e_2\|_\infty + \delta_R \|e_1\|_\infty \|s\|_\infty \\ &\leq \delta_R^2 \|u\|_\infty + \delta_R + \delta_R^2 \|s\|_\infty \\ &= \delta_R^2(\|u\|_\infty + \|s\|_\infty) + \delta_R.\end{aligned}$$

Since $\|s\|_\infty = \|u\|_\infty = 1$, $\|e_0\|_\infty \leq 2\delta_R^2 + \delta_R$. By assumption $p_0 > \frac{4\delta_R^2 + 2\delta_R}{\delta_R - 1}$, so $\texttt{BFV.Modreduce}(\texttt{ct}_0, p_0 q, q)$ outputs $\texttt{ct}_0' = (a_0', b_0^*)$ satisfying

$$b_0^* \equiv -a_0' s + e_0' \mod (\phi(x), q)$$

with $\|e_0'\|_\infty \leq \rho$ by Lemma 2.1. Since $b_0' = [b_0^* + D_q m_0]_q$, we have

$$b_0' + a_0' s \equiv D_q m_0 + e_0' \mod (\phi(x), q).$$

Proof of Lemma 3.2. By assumption, $b_0 + a_0 s \equiv D_q m_0 + e_0 \mod (\phi(x), q)$ with $\|e_0\|_\infty < (D_q - 1)/2$. There is a polynomial $r \in R_n$ such that $[b_0 + a_0 s]_{\phi(x),q} = D_q m_0 + e_0 + qr \mod \phi(x)$. Thus,

$$tc = t[b_0 + a_0 s]_{\phi(x),q} = tD_q m_0 + te_0 + tqr = qm_0 - m_0 + te_0 + tqr.$$

Then,

$$\Big[\Big\lfloor \frac{tc}{q} \Big\rceil\Big]_t = \Big[m_0 + \Big\lfloor -\frac{m_0}{q} + \frac{t}{q} e_0 \Big\rceil + tr\Big]_t = \Big[m_0 + \Big\lfloor \frac{t}{q}(e_0 - \frac{1}{t} m_0) \Big\rceil\Big]_t = m_0.$$

The last equality follows from the fact that $\Big\lfloor \frac{t}{q}(e_0 - \frac{1}{t} m_0) \Big\rceil = 0$, as $\|e_0\|_\infty < (D_q - 1)/2$.

Proof of Lemma 3.3. By assumption, each $\texttt{ct}_i = (a_i, b_i) \in R_{n,q}^2$ satisfies

$$b_i + a_i s \equiv D_q m_i + e_i \mod (\phi(x), q)$$

with $\|e_i\|_\infty \leq E$ for all $i = 0, \ldots, k-1$. Then,

$$\texttt{ct}_0' = \Big[\sum_{i=0}^{k-1} \alpha_i \texttt{ct}_i\Big]_q = \Big(\Big[\sum_{i=0}^{k-1} \alpha_i a_i\Big]_q, \Big[\sum_{i=0}^{k-1} \alpha_i b_i\Big]_q\Big).$$

Since $t|(q-1)$ and $D_q = (q-1)/t$, we have $D_q t \equiv -1 \mod q$. There exists $r \in R_n$ such that

$$\sum_{i=0}^{k-1} \alpha_i m_i = \Big[\sum_{i=0}^{k-1} \alpha_i m_i\Big]_t + tr$$

with $\|r\|_\infty \le M$. Then,

$$\begin{aligned}\sum_{i=0}^{k-1}\alpha_i b_i + \sum_{i=0}^{k-1}\alpha_i a_i s &\equiv D_q\Big(\sum_{i=0}^{k-1}\alpha_i m_i\Big) + \sum_{i=0}^{k-1}\alpha_i e_i \mod (\phi(x), q)\\ &\equiv D_q\Big[\sum_{i=0}^{k-1}\alpha_i m_i\Big]_t + \sum_{i=0}^{k-1}\alpha_i e_i + D_q t r \mod (\phi(x), q)\\ &\equiv D_q\Big[\sum_{i=0}^{k-1}\alpha_i m_i\Big]_t + \sum_{i=0}^{k-1}\alpha_i e_i - r \mod (\phi(x), q)\end{aligned}$$

Let $e' = \sum_{i=0}^{k-1}\alpha_i e_i - r$. Then, we have

$$\sum_{i=0}^{k-1}\alpha_i b_i + \sum_{i=0}^{k-1}\alpha_i a_i s \equiv D_q\Big[\sum_{i=0}^{k-1}\alpha_i m_i\Big]_t + e' \mod (\phi(x), q).$$

So, $\mathtt{ct}_0'$ is a BFV ciphertext with noise term e' and

$$\|e'\|_\infty \le \sum_{i=0}^{k-1}|\alpha_i|\,\|e_i\|_\infty + \|r\|_\infty \le ME + M = M(E+1).$$

Proof of Lemma 3.4. By assumption, we have for $i = 0, 1$,

$$b_i + a_i s \equiv D_q m_i + e_i \mod (\phi(x), q) \tag{16}$$

with $\|a_i\|_\infty \le q/2, \|b_i\|_\infty \le q/2$, and $\|e_i\|_\infty \le E$. We can rewrite (16) as

$$b_i + a_i s \equiv D_q m_i + e_i + q r_i \mod \phi(x) \tag{17}$$

where

$$\|r_i\|_\infty \le \frac{1}{q}\|b_i + a_i s\|_\infty \le \frac{1}{q}\Big(\frac{q}{2} + \frac{q}{2}\delta_R\|s\|_\infty\Big) \le \frac{1}{2}\Big(1 + \delta_R\|s\|_\infty\Big) \le \delta_R\|s\|_\infty.$$

On the one hand,

$$\begin{aligned}(t/q)(b_0 + a_0 s)(b_1 + a_1 s) &\equiv (t/q)\Big(b_0 b_1 + (b_1 a_0 + b_0 a_1)s + a_0 a_1 s^2\Big) \mod \phi(x)\\ &\equiv (t/q)(c_0 + c_1 s + c_2 s^2) \mod \phi(x)\\ &\equiv c_0' + c_1' s + c_2' s^2 + (\epsilon_0' + \epsilon_1' s + \epsilon_2' s^2) \mod \phi(x)\end{aligned}$$

with $\|\epsilon_i'\|_\infty \le 1/2$ for $0 \le i \le 2$. Let $r_m \in R_n$ so that $m_0 m_1 = [m_0 m_1]_{\phi(x),t} + t r_m$ with $\|r_m\|_\infty \le t\delta_R/4$. Then on the other hand, noting that $(t/q)D_q = (t/q)(q/t - 1/t) =$

$1 - 1/q$ and $tD_q \equiv -1 \mod q$, we have

$$
\begin{aligned}
&(t/q)(D_q m_0 + e_0 + q r_0)(D_q m_1 + e_1 + q r_1) \\
&\equiv (t/q)(D_q^2 m_0 m_1 + D_q(m_0 e_1 + m_1 e_0) \\
&\quad + q(e_0 r_1 + r_0 e_1) + e_0 e_1 + q D_q(m_0 r_1 + r_0 m_1) + q^2 r_0 r_1) \mod \phi(x) \\
&\equiv (1 - 1/q) D_q [m_0 m_1]_{\phi(x),t} + (1 - 1/q) D_q t r_m + (1 - 1/q)(m_0 e_1 + m_1 e_0) \\
&\quad + t(e_0 r_1 + r_0 e_1) + (t/q) e_0 e_1 + t D_q(m_0 r_1 + r_0 m_1) + t q r_0 r_1 \mod \phi(x) \\
&\equiv D_q [m_0 m_1]_{\phi(x),t} - (D_q/q)[m_0 m_1]_{\phi(x),t} + D_q t r_m \\
&\quad - (D_q t/q) r_m + (m_0 e_1 + m_1 e_0) - (1/q)(m_0 e_1 + m_1 e_0) \\
&\quad + t(e_0 r_1 + r_0 e_1) + (t/q) e_0 e_1 + t D_q(m_0 r_1 + r_0 m_1) + t q r_0 r_1 \mod \phi(x) \\
&\equiv D_q [m_0 m_1]_{\phi(x),t} - r_m + (m_0 e_1 + m_1 e_0) + t(e_0 r_1 + r_0 e_1) - (m_0 r_1 + r_0 m_1) \\
&\quad - (D_q/q)[m_0 m_1]_{\phi(x),t} - (1/q)(m_0 e_1 + m_1 e_0) + (t/q) e_0 e_1 \mod (\phi(x), q).
\end{aligned}
$$

Let

$$
\begin{aligned}
\epsilon_1 &= \epsilon_0' + \epsilon_1' s + \epsilon_2' s^2 \quad \text{and} \\
\epsilon_2 &= -r_m + (m_0 e_1 + m_1 e_0) + t(e_0 r_1 + r_0 e_1) - (m_0 r_1 + r_0 m_1) \\
&\quad - (D_q/q)[m_0 m_1]_{\phi(x),t} - (1/q)(m_0 e_1 + m_1 e_0) + (t/q) e_0 e_1.
\end{aligned}
$$

Let $e' = \epsilon_2 - \epsilon_1$. Then,

$$c_0' + c_1' s + c_2' s^2 \equiv D_q [m_0 m_1]_{\phi(x),t} + e' \mod (\phi(x), q).$$

We now turn to the noise bound for e'. First, note that

$$\|\epsilon_1\|_\infty = \left\| \epsilon_0' + \epsilon_1' s + \epsilon_2' s^2 \right\|_\infty \leq \frac{1}{2} + \frac{\delta_R \|s\|_\infty}{2} + \frac{\delta_R^2 \|s\|_\infty^2}{2} = \frac{(1 + \delta_R \|s\|_\infty)^2}{2}.$$

To simplify the noise analysis for ϵ_2, we break down ϵ_2 into two pieces. Let

$$
\begin{aligned}
\omega_1 &= -r_m + (m_0 e_1 + m_1 e_0) + t(e_0 r_1 + r_0 e_1) - (m_0 r_1 + r_0 m_1) \quad \text{and} \\
\omega_2 &= -(D_q/q)[m_0 m_1]_{\phi(x),t} - (D_q t/q) r_m - (1/q)(m_0 e_1 + m_1 e_0) + (t/q) e_0 e_1.
\end{aligned}
$$

Then, $\epsilon_2 = \omega_1 + \omega_2$. For ω_1, we have

$$
\begin{aligned}
\|\omega_1\|_\infty &\leq \|r_m\|_\infty + \|m_0 e_1 + m_1 e_0\|_\infty + t \|e_0 r_1 + r_0 e_1\|_\infty + \|m_0 r_1 + r_0 m_1\|_\infty \\
&\leq \frac{t\delta_R}{4} + E t \delta_R + 2 E t \delta_R^2 \|s\|_\infty + t \delta_R^2 \|s\|_\infty .
\end{aligned}
$$

For ω_2, note that since $D_q < q/t$ and $q > 2t$, we have

$$\begin{aligned}
\|\omega_2\|_\infty &\leq \frac{D_q}{q}\left\|[m_0m_1]_{\phi(x),t}\right\|_\infty + \frac{D_q t}{q}\|r_m\|_\infty + \frac{1}{q}\|m_0e_1 + m_1e_0\|_\infty + \frac{t}{q}\|e_0e_1\|_\infty \\
&< \frac{1}{t}\left\|[m_0m_1]_{\phi(x),t}\right\|_\infty + \|r_m\|_\infty + \frac{1}{q}\|m_0e_1 + m_1e_0\|_\infty + \frac{t}{q}\|e_0e_1\|_\infty \\
&\leq \frac{1}{2} + \frac{t\delta_R}{4} + \frac{Et\delta_R}{q} + \frac{E\delta_R}{2} \\
&\leq \frac{1}{2} + \frac{t\delta_R}{4} + E\delta_R.
\end{aligned}$$

The bound on $\frac{t}{q}\|e_0e_1\|_\infty$ follows from the fact that since e_0, e_1 are ciphertext noise terms, so by assumption $\|e_i\|_\infty \leq E < (D_q - 1)/2 < D_q/2 < q/(2t)$ for $i = 0, 1$. So,

$$\frac{t}{q}\|e_0e_1\|_\infty \leq \frac{t}{q}E^2\delta_R < \frac{E\delta_R}{2}.$$

By assumption, $\delta_R \geq 16$, $E \geq 1$, $t \geq 2$, and $\|s\|_\infty = 1$. So,

$$\begin{aligned}
\left\|e'\right\|_\infty &\leq \frac{t\delta_R}{4} + Et\delta_R + 2Et\delta_R^2\|s\|_\infty + t\delta_R^2\|s\|_\infty + \frac{1}{2} + \frac{t\delta_R}{4} + E\delta_R + \frac{(\delta_R\|s\|_\infty + 1)^2}{2} \\
&\leq \frac{t\delta_R}{2} + Et\delta_R + 3Et\delta_R^2\|s\|_\infty + 1 + E\delta_R + \frac{\delta_R^2\|s\|_\infty^2}{2} + \delta_R\|s\|_\infty \\
&= Et\delta_R^2\|s\|_\infty^2\left(\frac{1}{E\delta_R\|s\|_\infty^2} + \frac{1}{\delta_R^2\|s\|_\infty^2} + \frac{3}{\|s\|_\infty} + \frac{1}{Et\delta_R^2\|s\|_\infty^2} + \frac{1}{t\delta_R\|s\|_\infty^2} + \frac{1}{2Et} + \frac{1}{Et\delta_R\|s\|_\infty}\right) \\
&\leq Et\delta_R^2\|s\|_\infty^2\left(\frac{1}{16} + \frac{1}{16^2} + 3 + \frac{1}{2\cdot 16^2} + \frac{1}{2\cdot 16} + \frac{1}{2\cdot 2} + \frac{1}{2\cdot 16}\right) \\
&< 3.5Et\delta_R^2\|s\|_\infty^2 \\
&= 3.5tE\rho^2.
\end{aligned}$$

Proof of Lemma 3.5. Notice that as $\beta_1 \equiv c_2'\mathtt{k}_1' \mod (\phi(x), p_1q)$, we can write

$$\beta_1 \equiv c_2'\mathtt{k}_1' + p_1q\alpha_1 \mod \phi(x)$$

for some $\alpha_1 \in \mathbb{Z}$. Then,

$$d_1' = \left\lfloor \frac{\beta_1}{p_1} \right\rceil \equiv \left\lfloor \frac{c_2'\mathtt{k}_1' + p_1q\alpha_1}{p_1} \right\rceil \equiv \frac{c_2'\mathtt{k}_1'}{p_1} + q\alpha_1 + \epsilon_1 \equiv \frac{c_2'\mathtt{k}_1'}{p_1} + \epsilon_1 \mod (\phi(x), q)$$

for some $\epsilon_1 \in \mathbb{R}[x]/(\phi(x))$ with $\|\epsilon_1\|_\infty \leq 1/2$. Note also that as $c_2'\mathtt{k}_0' \equiv -c_2'\mathtt{k}_1's + c_2'e_1' + c_2'p_1s^2 \mod (\phi(x), p_1q)$, we have that for some $\alpha_0 \in \mathbb{Z}$,

$$\beta_0 \equiv c_2'\mathtt{k}_0' \equiv -c_2'\mathtt{k}_1's + c_2'e_1' + c_2'p_1s^2 + p_1q\alpha_0 \mod \phi(x).$$

Then,

$$\begin{aligned}
d'_0 &= \left\lfloor \frac{\beta_0}{p_1} \right\rceil \\
&\equiv \left\lfloor \frac{-c'_2\mathtt{k}'_1 s + c'_2 e'_1 + c'_2 p_1 s^2 + p_1 q\alpha_0}{p_1} \right\rceil \mod (\phi(x), q) \\
&\equiv \left\lfloor \frac{-c'_2\mathtt{k}'_1 s}{p_1} + \frac{c'_2 e'_1}{p_1} + c'_2 s^2 + q\alpha_0 \right\rceil \mod (\phi(x), q) \\
&\equiv \frac{-c'_2\mathtt{k}'_1 s}{p_1} + \frac{c'_2 e'_1}{p_1} + c'_2 s^2 + q\alpha_0 + \epsilon_0 \mod (\phi(x), q) \\
&\equiv \frac{-c'_2\mathtt{k}'_1 s}{p_1} + \frac{c'_2 e'_1}{p_1} + c'_2 s^2 + \epsilon_0 \mod (\phi(x), q)
\end{aligned}$$

for some $\epsilon_0 \in \mathbb{R}[x]/(\phi(x))$ with $\|\epsilon_0\|_\infty \leq 1/2$. Therefore,

$$\begin{aligned}
d'_0 + d'_1 s &\equiv c'_2 s^2 - \frac{c'_2\mathtt{k}'_1 s}{p_1} + \frac{c'_2 e'_1}{p_1} + \epsilon_0 + \frac{c'_2\mathtt{k}'_1 s}{p_1} + \epsilon_1 s \mod (\phi(x), q) \\
&\equiv c'_2 s^2 + \frac{c'_2 e'_1}{p_1} + \epsilon_0 + \epsilon_1 s \mod (\phi(x), q)
\end{aligned}$$

Let $e'' = \frac{c'_2 e'_1}{p_1} + \epsilon_0 + \epsilon_1 s$. Then, $d'_0 + d'_1 s \equiv c'_2 s^2 + e'' \mod (\phi(x), q)$. By assumption, (c'_0, c'_1, c'_2) satisfies

$$c'_0 + c'_1 s + c'_2 s^2 \equiv D_q[m_0 m_1]_{\phi(x),t} + e' \mod (\phi(x), q).$$

Let $e^* = e' + e''$. Then,

$$\begin{aligned}
c_1 + c_0 s &\equiv c'_0 + d'_0 + c'_1 s + d'_1 s \mod (\phi(x), q) \\
&\equiv c'_0 + c'_1 s + c'_2 s^2 + e'' \mod (\phi(x), q) \\
&\equiv D_q[m_0 m_1]_{\phi(x),t} + e' + e'' \mod (\phi(x), q) \\
&\equiv D_q[m_0 m_1]_{\phi(x),t} + e^* \mod (\phi(x), q)
\end{aligned}$$

Now, we turn to the noise bounds on e'' and e^*. Note that as $e'_1 \leftarrow \chi_\rho$, we have $\left\|e'_1\right\|_\infty \leq \rho = \delta_R \|s\|_\infty$. If $p_1 \geq 6q$ and $\delta_R \geq 16$, then

$$\begin{aligned}
\left\|e''\right\|_\infty &= \left\| \frac{c'_2 e'_1}{p_1} + \epsilon_0 + \epsilon_1 s \right\|_\infty \\
&\leq \frac{q}{2p_1}\delta_R^2 \|s\|_\infty + \frac{1}{2} + \frac{\delta_R \|s\|_\infty}{2} \\
&\leq \delta_R^2 \|s\|_\infty \left(\frac{1}{12} + \frac{1}{\delta_R^2 \|s\|_\infty} + \frac{1}{2\delta_R}\right) \\
&< \frac{1}{8}\delta_R^2 \|s\|_\infty .
\end{aligned}$$

By Lemma 7, we have that $\left\|e'\right\|_\infty \leq 3.5Et\rho^2 = 3.5Et\delta_R^2 \left\|s\right\|_\infty^2$. As relinearization introduces additional noise e'' bounded by $\frac{1}{8}\delta_R^2 \left\|s\right\|_\infty$, we have

$$\begin{aligned}
\left\|e^*\right\|_\infty &\leq 3.5Et\delta_R^2 \left\|s\right\|_\infty^2 + \frac{1}{8}\delta_R^2 \left\|s\right\|_\infty \\
&= Et\delta_R^2 \left\|s\right\|_\infty^2 \left(3.5 + \frac{1}{8Et \left\|s\right\|_\infty}\right) \\
&\leq Et\delta_R^2 \left\|s\right\|_\infty^2 \left(3.5 + \frac{1}{16}\right) \\
&< 3.6Et\delta_R^2 \left\|s\right\|_\infty^2 \\
&= 3.6Et\rho^2.
\end{aligned}$$

Proof of Lemma 3.6. By assumption, the public key $\mathtt{pk} = (\mathtt{k}_0, \mathtt{k}_1)$ satisfies

$$\mathtt{k}_1 + \mathtt{k}_0 s \equiv te \mod (\phi(x), p_0 q)$$

for some noise $e \in R_n$ with $\left\|e\right\|_\infty \leq \rho$. Then,

$$\begin{aligned}
b_0 + a_0 s &\equiv \mathtt{k}_1 u + te_2 + (\mathtt{k}_0 u + te_1)s \mod (\phi(x), p_0 q) \\
&\equiv -(\mathtt{k}_0 s + te)u + te_2 + (\mathtt{k}_0 u + te_1)s \mod (\phi(x), p_0 q) \\
&\equiv -\mathtt{k}_0 su - teu + te_2 + \mathtt{k}_0 su + te_1 s \mod (\phi(x), p_0 q) \\
&\equiv -t(eu + e_2 + e_1 s) \mod (\phi(x), p_0 q) \\
&\equiv te_0 \mod (\phi(x), p_0 q)
\end{aligned}$$

Let $e_0 = -eu + e_2 + e_1 s$. Then, $b_0 + a_0 s \equiv te_0 \mod (\phi(x), p_0 q)$. Since $\left\|s\right\|_\infty = \left\|u\right\|_\infty = 1$, we have $\left\|e_0\right\|_\infty \leq 2\delta_R^2 + \delta_R$. By assumption $p_0 > \frac{4\delta_R^2 + 2\delta_R}{\delta_R - 2}$, so $\mathtt{BGV.Modreduce}(\mathtt{ct}_0, p_0 q, q)$ outputs $\mathtt{ct}_0' = (a_0', b_0^*)$ satisfying

$$b_0^* \equiv -a_0' s + te_0' \mod (\phi(x), q)$$

with $\left\|e_0'\right\|_\infty \leq \rho$ by Lemma 2.3. Since $b_0' = [b_0^* + m_0]_q$, we have

$$b_0' + a_0' s \equiv m_0 + te_0' \mod (\phi(x), q).$$

Proof of Lemma 3.7. By assumption, we have for $i = 0, 1$,

$$b_i + a_i s \equiv m_i + te_i \mod (\phi(x), q)$$

with $\left\|e_i\right\|_\infty \leq E$. Let $r_m \in R_n$ such that $m_0 m_1 = [m_0 m_1]_{\phi(x),t} + tr_m$. Note that $\left\|r_m\right\|_\infty \leq t$. We have that

$$\begin{aligned}
c_0' + c_1' s + c_2' s^2 &\equiv b_0 b_1 + (b_1 a_0 + b_0 a_1)s + a_0 a_1 s^2 \mod (\phi(x), q) \\
&\equiv (b_0 + a_0 s)(b_1 + a_1 s) \mod (\phi(x), q) \\
&\equiv (m_0 + te_0)(m_1 + te_1) \mod (\phi(x), q) \\
&\equiv m_0 m_1 + t(m_0 e_1 + m_1 e_0 + te_0 e_1) \mod (\phi(x), q) \\
&\equiv [m_0 m_1]_{\phi(x),t} + t(m_0 e_1 + m_1 e_0 + te_0 e_1 + r_m) \mod (\phi(x), q)
\end{aligned}$$

Let $e' = m_0 e_1 + m_1 e_0 + t e_0 e_1 + r_m$. Then,

$$c'_0 + c'_1 s + c'_2 s^2 \equiv [m_0 m_1]_{\phi(x),t} + te' \mod (\phi(x), q)$$

with

$$\left\|e'\right\|_\infty \leq \delta_R t E + \delta_R t E^2 + t \leq 2\delta_R t E^2 + t \leq 2\delta_R t (E^2 + 1).$$

Proof of Lemma 3.8. First, observe that both $\beta_0 + t\omega_0$ and $\beta_1 + t\omega_1$ are divisible by p_1 since for $i = 0, 1$,

$$\beta_i + t\omega_i = \beta_i + t[-t^{-1}\beta_i]_{p_1} \equiv \beta_i + t(-t^{-1}\beta_i) \equiv 0 \mod p_1,$$

so $d'_0 = \frac{\beta_0 + t\omega_0}{p_1}$ and $d'_1 = \frac{\beta_1 + t\omega_1}{p_1}$ have integer coefficients. Then,

$$\begin{aligned}
d'_0 + d'_1 s &\equiv \frac{\beta_0 + t\omega_0}{p_1} + \frac{\beta_1 + t\omega_1}{p_1} s \mod (\phi(x), q) \\
&\equiv \frac{c'_2 \mathtt{k}'_0 + \omega_0}{p_1} + \frac{c'_2 \mathtt{k}'_1 + \omega_1}{p_1} s \mod (\phi(x), q) \\
&\equiv \frac{c'_2(-\mathtt{k}'_1 s + p_1 s^2 + te'_1) + t\omega_0}{p_1} + \frac{c'_2 \mathtt{k}'_1 + t\omega_1}{p_1} s \mod (\phi(x), q) \\
&\equiv c'_2 s^2 + t\frac{c'_2 e'_1 + \omega_0 + \omega_1 s}{p_1} \mod (\phi(x), q).
\end{aligned}$$

Let $e'' = \frac{c'_2 e'_1}{p_1} + \frac{\omega_0 + \omega_1 s}{p_1}$, which has integer coefficients. Then, $d'_0 + d'_1 s \equiv c'_2 s^2 + te''$. By assumption, (c'_0, c'_1, c'_2) satisfies

$$c'_0 + c'_1 s + c'_2 s^2 \equiv [m_0 m_1]_{\phi(x),t} + te' \mod (\phi(x), q),$$

where $\left\|e'\right\|_\infty \leq 2\delta_R t (E^2 + 1)$. Let $e^* = e' + e''$. Then,

$$\begin{aligned}
c_1 + c_0 s &\equiv c'_0 + d'_0 + c'_1 s + d'_1 s \mod (\phi(x), q) \\
&\equiv c'_0 + c'_1 s + c'_2 s^2 + te'' \mod (\phi(x), q) \\
&\equiv [m_0 m_1]_{\phi(x),t} + t(e' + e'') \mod (\phi(x), q) \\
&\equiv [m_0 m_1]_{\phi(x),t} + te^* \mod (\phi(x), q).
\end{aligned}$$

Then, note that

$$\begin{aligned}
\left\|e''\right\|_\infty &\leq \left\|\frac{c_2' e_1'}{p_1}\right\|_\infty + \left\|\frac{\omega_0 + \omega_1 s}{p_1}\right\|_\infty \\
&\leq \frac{q}{2p_1}\delta_R^2 \left\|s\right\|_\infty + \frac{p_1}{2p_1} + \frac{p_1}{2p_1}\delta_R \left\|s\right\|_\infty \\
&\leq \frac{1}{12}\delta_R^2 \left\|s\right\|_\infty + \frac{1}{2} + \frac{\delta_R \left\|s\right\|_\infty}{2} \\
&= \delta_R^2 \left\|s\right\|_\infty (\frac{1}{12} + \frac{1}{2\delta_R^2 \left\|s\right\|_\infty} + \frac{1}{2\delta_R}) \\
&\leq \delta_R^2 \left\|s\right\|_\infty (\frac{1}{12} + \frac{1}{512} + \frac{1}{32}) \\
&< \frac{1}{8}\delta_R^2 \left\|s\right\|_\infty .
\end{aligned}$$

The bound on e^* then follows immediately.

Proof of Lemma 3.9. By assumption, we first note that $(a_0, b_0) \in R_{n,Q}^2$ satisfies

$$b_0 \equiv -a_0 s + m_0 + e_0 \mod (\phi(x), Q).$$

Therefore, there is some integer $r \in \mathbb{Z}$ such that $b_0 + a_0 s \equiv m_0 + e_0 + Qr \mod \phi(x)$. Let $\epsilon_1 = qa_0/Q - a_0'$, and $\epsilon_2 = qb_0/Q - b_0'$. Then,

$$\begin{aligned}
b_0' &= \frac{qb_0}{Q} - \epsilon_2 \\
&\equiv -\frac{qa_0 s}{Q} + \frac{q}{Q}m_0 + \frac{qe_0}{Q} - \epsilon_2 + qr \mod \phi(x) \\
&\equiv -a_0' s - \epsilon_1 s + \frac{q}{Q}m_0 + \frac{qe_0}{Q} - \epsilon_2 + qr \mod \phi(x)
\end{aligned}$$

Let $e_0' = \frac{q}{Q}e_0 - \epsilon_2 - \epsilon_1 s$. Then, $b_0' + a_0' s \equiv \frac{q}{Q}m_0 + e_0' \mod (\phi(x), q)$. Therefore,

$$\left\|e_0'\right\|_\infty \leq \frac{q}{Q}E + \frac{1 + \delta_R \left\|s\right\|_\infty}{2}.$$

By assumption $Q/q > \frac{2E}{\delta_R \left\|s\right\|_\infty - 1}$, so $\left\|e_0'\right\|_\infty < \delta_R \left\|s\right\|_\infty$.

Proof of Lemma 3.10. We consider step 2 of Algorithm 14, in which we compute $\texttt{ct} := \texttt{BFV.Encrypt}(m, 1, \texttt{pk})$. By an argument identical to the proof of Lemma 3.1, steps 1-3 of Algorithm 4 produce an ordered pair $(a_0, b_0) \in R_{n,p_0 q}^2$ satisfying

$$b_0 + a_0 s \equiv e_0 \mod (\phi(x), p_0 q)$$

for $e_0 \in R_n$ with $\|e_0\|_\infty \leq 2\delta_R^2 + \delta_R$. By assumption $p_0 > \frac{4\delta_R^2+2\delta_R}{\delta_R-1}$, so $(a_0', b_0^*) := \texttt{CKKS.Modreduce}(p_0 q, q, \texttt{ct}_0)$ satisfies

$$b_0^* \equiv -a_0' s + e_0' \mod (\phi(x), q)$$

with $\|e_0'\|_\infty \leq \rho$ by Lemma 3.9. Since $b_0' = [b_0^* + m]_q$, we have

$$b_0' + a_0' s \equiv m + e_0' \mod (\phi(x), q).$$

Let $(a, b) = (a_0', b_0')$. Then, $\texttt{ct} = (a, b)$ is the output of Algorithm 14 and is a CKKS ciphertext with noise bounded by ρ.

Proof of Lemma 3.11. By assumption, each $\texttt{ct}_i = (a_i, b_i) \in R_{n,q}^2$ satisfies

$$b_i + a_i s \equiv m_i + e_i \mod (\phi(x), q)$$

for some noise term e_i with $\|e_i\|_\infty \leq E$. Then,

$$\texttt{ct}_0' = \Big[\sum_{i=0}^{k-1} \alpha_i \texttt{ct}_i\Big]_q = \Big(\Big[\sum_{i=0}^{k-1} \alpha_i a_i\Big]_q, \Big[\sum_{i=0}^{k-1} \alpha_i b_i\Big]_q\Big).$$

Let $e' = \sum_{i=0}^{k-1} \alpha_i e_i$. Then,

$$\sum_{i=0}^{k-1} \alpha_i b_i + \sum_{i=0}^{k-1} \alpha_i a_i s \equiv \sum_{i=0}^{k-1} \alpha_i m_i + \sum_{i=0}^{k-1} \alpha_i e_i \equiv \sum_{i=0}^{k-1} \alpha_i m_i + e' \mod (\phi(x), q)$$

So, $\texttt{ct}_0'$ is a CKKS ciphertext with noise term e' and

$$\|e'\|_\infty \leq \sum_{i=0}^{k-1} |\alpha_i| \, \|e_i\|_\infty \leq ME.$$

Proof of Lemma 3.12. By assumption, we have for $i = 0, 1$,

$$b_i + a_i s \equiv m_i + e_i \mod (\phi(x), q)$$

with $\|e_i\|_\infty \leq E$. Let $e' = m_0 e_1 + m_1 e_0 + e_0 e_1$. Then,

$$\begin{aligned} c_0' + c_1' s + c_2' s^2 &= (b_0 + a_0 s)(b_1 + a_1 s) \\ &\equiv (m_0 + e_0)(m_1 + e_1) \mod (\phi(x), q) \\ &\equiv m_0 m_1 + m_0 e_1 + m_1 e_0 + e_0 e_1 \mod (\phi(x), q) \\ &\equiv m_0 m_1 + e' \mod (\phi(x), q) \end{aligned}$$

The bound on e' then follows immediately.

Proof of Lemma 3.13. It is clear from the proof of Lemma 3.5 that $d'_0+d'_1 s \equiv c'_2 s^2+ e'' \mod (\phi(x), q)$ where $e'' = \frac{c'_2 e'_1}{p_1} + \epsilon_0 + \epsilon_1 s$ and $\epsilon_0, \epsilon_1 \in \mathbb{R}[x]/(\phi(x))$ with $\|\epsilon_0\|_\infty \leq 1/2$ and $\|\epsilon_1\|_\infty \leq 1/2$. By assumption, $c'_0 + c'_1 s + c'_2 s^2 \equiv m_0 m_1 + e' \mod (\phi(x), q)$ with $\|e'\|_\infty \leq Et\delta_R + E^2\delta_R$, so

$$\begin{aligned} c_1 + c_0 s &\equiv c'_0 + d'_0 + c'_1 s + d'_1 s \mod (\phi(x), q) \\ &\equiv c'_0 + c'_1 s + c'_2 s^2 + e'' \mod (\phi(x), q) \\ &\equiv m_0 m_1 + e' + e'' \mod (\phi(x), q) \end{aligned}$$

Let $e^* = e' + e''$. Then, $c_1 + c_0 s \equiv m_0 m_1 + e^* \mod (\phi(x), q)$. The bound on e^* follows immediately.

Proof of Lemma 4.1. Suppose we have a collection of BFV ciphertexts, each with noise bounded by $\rho = \delta_R \|s\|_\infty$. By Lemma 3.3, computing

$$\mathtt{ct}_j := \mathtt{Linearcombo}(\mathtt{ct}_{j,1}, \ldots, \mathtt{ct}_{j,k_1}, 1, \ldots, 1)$$

results in BFV ciphertexts $\mathtt{ct}_j$ for $j = 1, \ldots, 2k_2$, each with noise bounded by $k_1\delta_R \|s\|_\infty + k_1$. Since $\delta_R \geq 16$, we have

$$k_1\delta_R \|s\|_\infty + k_1 = (k_1 + \frac{k_1}{\delta_R \|s\|_\infty})\delta_R \|s\|_\infty \leq \frac{17}{16} k_1 \delta_R \|s\|_\infty .$$

By Lemma 3.4, each $\mathtt{Multiply}(\mathtt{ct}_{2j-1}, \mathtt{ct}_{2j})$ in step 2 of Algorithm 17 for $j = 1, \ldots, k_2$ results in a polynomial triple with noise bounded by

$$3.5\Big(\frac{17}{16} k_1 \delta_R \|s\|_\infty\Big) t\delta_R^2 \|s\|_\infty^2 = \frac{119}{32} k_1 t \delta_R^3 \|s\|_\infty^3 .$$

Summing all k_2 of these polynomial triples in step 2 of Algorithm 17 results in a polynomial triple with noise bounded by

$$\frac{119}{32} k_1 k_2 t \delta_R^3 \|s\|_\infty^3 + k_2$$

by an equivalent argument to Lemma 3.3 with polynomial triples as the input. Notice,

$$\begin{aligned} \frac{119}{32} k_1 k_2 t \delta_R^3 \|s\|_\infty^3 + k_2 &= \frac{119}{32} k_1 k_2 t \delta_R^3 \|s\|_\infty^3 \left(1 + \frac{32}{119 k_1 t \delta_R^3 \|s\|_\infty^3}\right) \\ &\leq \frac{119}{32} k_1 k_2 t \delta_R^3 \|s\|_\infty^3 \left(1 + \frac{32}{119 \cdot 2 \cdot 16^3}\right) \\ &\leq \frac{30}{8} k_1 k_2 t \delta_R^3 \|s\|_\infty^3 . \end{aligned}$$

By the proof of Lemma 3.5, `BFV.Relinearize` introduces additional noise of at most $\frac{1}{8}\delta_R^2 \|s\|_\infty$. So, after performing relinearization in step 3 of Algorithm 17 we have a BFV ciphertext with noise bounded by

$$\begin{aligned}\frac{30}{8}k_1k_2t\delta_R^3\|s\|_\infty^3 + \frac{1}{8}\delta_R^2\|s\|_\infty &= \frac{30}{8}k_1k_2t\delta_R^3\|s\|_\infty^3\left(1+\frac{1}{30k_1k_2t\delta_R\|s\|_\infty^2}\right)\\ &\leq \frac{30}{8}k_1k_2t\delta_R^3\|s\|_\infty^3\left(1+\frac{1}{30\cdot 2\cdot 16}\right)\\ &\leq \frac{31}{8}k_1k_2t\delta_R^3\|s\|_\infty^3.\end{aligned}$$

So, a worst case noise bound for a depth-1 multiplication is given by $\frac{31}{8}k_1k_2t\delta_R^3\|s\|_\infty^3$. Since $\delta_R - 2 \geq \frac{7}{8}\delta_R$ and $\delta_R\|s\|_\infty - 2 \geq \frac{7}{8}\delta_R\|s\|_\infty$, we have

$$\frac{2(\frac{31}{8}k_1k_2t\delta_R^3\|s\|_\infty^3)}{\delta_R\|s\|_\infty - 2} \leq \frac{\frac{62}{8}k_1k_2t\delta_R^3\|s\|_\infty^3}{\frac{7}{8}\delta_R\|s\|_\infty} < 9k_1k_2t\delta_R^2\|s\|_\infty^2.$$

As $\delta_R = n$ and $\|s\|_\infty = 1$, we have that $9k_1k_2t\delta_R^2\|s\|_\infty^2 = 9k_1k_2tn^2 < q_i$. By Lemma 2.2, BFV modulus reduction from Q_i to Q_{i-1} gives a new ciphertext with noise bounded by ρ. Thus, the lemma is proved.

Proof of Lemma 4.2. Suppose we have a collection of BGV ciphertexts, each with noise bounded by $\rho = \delta_R\|s\|_\infty$. By a similar argument to Lemma 3.3, computing

$$\mathtt{ct}_j := \mathtt{Linearcombo}(\mathtt{ct}_{j,1},\ldots,\mathtt{ct}_{j,k_1},1,\ldots,1)$$

results in BGV ciphertexts $\mathtt{ct}_j$ for $j = 1,\ldots,2k_2$, each with noise bounded by $k_1\delta_R\|s\|_\infty + k_1$. Since $\delta_R \geq 16$, we have

$$k_1\delta_R\|s\|_\infty + k_1 = (k_1 + \frac{k_1}{\delta_R\|s\|_\infty})\delta_R\|s\|_\infty \leq \frac{17}{16}k_1\delta_R\|s\|_\infty.$$

By Lemma 3.7, each $\mathtt{Multiply}(\mathtt{ct}_{2j-1},\mathtt{ct}_{2j})$ in step 2 of Algorithm 17 for $j = 1,\ldots,k_2$ results in a polynomial triple with noise bounded by

$$2t\delta_R\Big(\Big(\frac{17}{16}k_1\delta_R\|s\|_\infty\Big)^2 + 1\Big) = 2t\delta_R\Big(\frac{289}{256}k_1^2\delta_R^2\|s\|_\infty^2 + 1\Big).$$

Then,

$$\begin{aligned}2t\delta_R\Big(\frac{289}{256}k_1^2\delta_R^2\|s\|_\infty^2 + 1\Big) &= \frac{289}{128}tk_1^2\delta_R^3\|s\|_\infty^2\left(1+\frac{256}{289k_1^2\delta_R^2\|s\|_\infty^2}\right)\\ &\leq \frac{289}{128}tk_1^2\delta_R^3\|s\|_\infty^2\left(1+\frac{1}{289}\right)\\ &= \frac{290}{128}tk_1^2\delta_R^3\|s\|_\infty^2.\end{aligned}$$

Summing all k_2 of these polynomial triples in step 2 of Algorithm 17 results in a polynomial triple with noise bounded by

$$\frac{290}{128}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3+k_2$$

Notice,

$$\begin{aligned}\frac{290}{128}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3+k_2&=\frac{290}{128}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3\left(1+\frac{128}{290k_1^2t\delta_R^3\left\|s\right\|_\infty^3}\right)\\&=\frac{290}{128}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3\left(1+\frac{128}{290\cdot 2\cdot 16^3}\right)\\&\leq\frac{290}{128}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3\left(1+\frac{1}{18560}\right)\\&\leq\frac{13}{8}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3.\end{aligned}$$

By the proof of Lemma 3.8, `BGV.Relinearize` introduces additional noise of at most $\frac{1}{8}\delta_R^2\left\|s\right\|_\infty^2$. So, after performing relinearization in step 3 of Algorithm 17 we have a BGV ciphertext with noise bounded by

$$\begin{aligned}\frac{13}{8}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3+\frac{1}{8}\delta_R^2\left\|s\right\|_\infty^2&=\frac{13}{8}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3\left(1+\frac{1}{13k_1^2k_2t\delta_R\left\|s\right\|_\infty^2}\right)\\&\leq\frac{13}{8}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3\left(1+\frac{1}{416}\right)\\&\leq\frac{14}{8}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3.\end{aligned}$$

So, a worst case noise bound for a depth-1 multiplication is given by $\frac{14}{8}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3$. Since $\delta_R-2\geq\frac{7}{8}\delta_R$ and $\delta_R\left\|s\right\|_\infty-2\geq\frac{7}{8}\delta_R\left\|s\right\|_\infty$, we have

$$\frac{2(\frac{14}{8}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3)}{\delta_R\left\|s\right\|_\infty-2}\leq\frac{\frac{28}{8}k_1^2k_2t\delta_R^3\left\|s\right\|_\infty^3}{\frac{7}{8}\delta_R\left\|s\right\|_\infty}=4k_1^2k_2t\delta_R^2\left\|s\right\|_\infty^2.$$

As $\delta_R=n$ and $\left\|s\right\|_\infty=1$, we have that $4k_1^2k_2t\delta_R^2\left\|s\right\|_\infty^2=4k_1^2k_2tn^2<q_i$. By Lemma 2.3, BGV modulus reduction from Q_i to Q_{i-1} gives a new ciphertext with noise bounded by ρ. Thus, the lemma is proved.

Proof of Lemma 4.3. Suppose we have a collection of CKKS ciphertexts, each with noise bounded by E. Recall that $\delta_R\left\|s\right\|_\infty=n$. By Lemma 3.11, computing

$$\mathtt{ct}_j:=\mathtt{Linearcombo}(\mathtt{ct}_{j,1},\ldots,\mathtt{ct}_{j,k_1},1,\ldots,1)$$

results in CKKS ciphertexts $\mathtt{ct}_j$ for $j = 1, \ldots, 2k_2$, each with noise bounded by $k_1 E$. Let $t = 2\Delta Z + 1$, and observe that for each encoding m_j of message z_j,

$$\left\|m_j\right\|_\infty \leq \Delta \left\|z_j\right\|_\infty + \frac{1}{2} \leq \Delta Z + \frac{1}{2} = t/2.$$

By Lemma 3.12, each $\mathtt{Multiply}(\mathtt{ct}_{2j-1}, \mathtt{ct}_{2j})$ in Step 2 of Algorithm 17 for $j = 1, \ldots, k_2$ results in a polynomial triple with noise bounded by

$$k_1 E t n + k_1^2 E^2 n = k_1 E(2\Delta Z + 1)n + k_1^2 E^2 n.$$

Summing all k_2 of these polynomial triples in Step 2 of Algorithm 17 results in a polynomial triple with noise bounded by

$$k_1 k_2 E(2\Delta Z + 1)n + k_1^2 k_2 E^2 n$$

by an equivalent argument to Lemma 3.11 with polynomial triples as the input. By the proof of Lemma 3.13, `CKKS.Relinearize` introduces additional noise of at most $\frac{1}{8}n^2$. So, after performing relinearization in Step 3 of Algorithm 17 we have a CKKS ciphertext with noise bounded by

$$k_1 k_2 E(2\Delta Z + 1)n + k_1^2 k_2 E^2 n + \frac{1}{8}n^2.$$

So, a worst-case noise bound for a depth-1 multiplication is given by $k_1 k_2 E(2\Delta Z + 1)n + k_1^2 k_2 E^2 n + \frac{1}{8}n^2$. Performing a rescaling operation then gives noise bounded by

$$\frac{2k_1 k_2 E \Delta Z n}{\Delta} + \frac{k_1 k_2 E n}{\Delta} + \frac{k_1^2 k_2 E^2 n}{\Delta} + \frac{n^2}{8\Delta} = 2k_1 k_2 n E Z + \frac{k_1 k_2 E n}{\Delta} + \frac{k_1^2 k_2 E^2 n}{\Delta} + \frac{1}{8}.$$

Proof of Lemma 4.7. By assumption the input of Algorithm 24 is the RNS representation in basis $\mathcal{B}$ of some $(c_0, c_1, c_2) \in R_{n,Q_i}^3$ satisfying

$$c_0 + c_1 s + c_2 s^2 \equiv D_{Q_i}[m_0 m_1]_{\phi(x),t} + e' \mod (\phi(x), Q_i).$$

for $\left\|e'\right\|_\infty \leq E$. From Step 1, $(\tilde{c}_2^{(0)}, \ldots, \tilde{c}_2^{(k-1)}, c_2^{(0)}, \ldots, c_2^{(i)})$ is the RNS representation of $\tilde{c}_2 \in R_{n,PQ_i}$ in basis $\mathcal{D}$ satisfying $\tilde{c}_2 = c_2 + Q_i e$ for some $e \in R_n$ with $\left\|\tilde{c}_2\right\|_\infty \leq Q_i(i+1)/2$ by Lemma 4.4. After Step 3, note $(\hat{a}^{(j)}, \hat{b}^{(j)})_{0 \leq j \leq i+k}$ is the RNS representation of some $(\hat{a}, \hat{b}) \in R_{n,PQ_i}^2$ satisfying

$$\begin{aligned}
\hat{b} + \hat{a}s &\equiv \tilde{c}_2 \tilde{\mathtt{k}}_1 + \tilde{c}_2 \tilde{\mathtt{k}}_0 s \mod (\phi(x), PQ_i) \\
&\equiv \tilde{c}_2(-\tilde{\mathtt{k}}_0 s + Ps^2 + \tilde{e}) + \tilde{c}_2 \tilde{\mathtt{k}}_0 s \mod (\phi(x), PQ_i) \\
&\equiv (c_2 + Q_i e)(-\tilde{\mathtt{k}}_0 s + Ps^2 + \tilde{e}) + (c_2 + Q_i e)\tilde{\mathtt{k}}_0 s \mod (\phi(x), PQ_i) \\
&\equiv (c_2 + Q_i e)(Ps^2 + \tilde{e}) \mod (\phi(x), PQ_i) \\
&\equiv c_2 P s^2 + c_2 \tilde{e} + PQ_i e s^2 + Q_i e \tilde{e} \mod (\phi(x), PQ_i) \\
&\equiv c_2 P s^2 + c_2 \tilde{e} + Q_i e \tilde{e} \mod (\phi(x), PQ_i) \\
&\equiv c_2 P s^2 + \hat{e} \mod (\phi(x), PQ_i)
\end{aligned}$$

for $\hat{e} = c_2\tilde{e} + Q_i e\tilde{e} = \tilde{c}_2\tilde{e}$. Note that as $\tilde{e} \leftarrow \chi_\rho$, we have $\|\hat{e}\|_\infty = \|\tilde{c}_2\tilde{e}\|_\infty \leq Q_i(i+1)\rho^2/2$. Furthermore, there exists $\omega \in R_n$ such that

$$\hat{b} + \hat{a}s \equiv c_2 P s^2 + \hat{e} + \omega P Q_i \mod \phi(x).$$

By Lemma 4.5, Step 4 returns the RNS representation of some $\hat{c}_0 \in R_{n,Q_i}$ and $\hat{c}_1 \in R_{n,Q_i}$ satisfying

$$\hat{c}_0 = \frac{\hat{b}}{P} + \hat{e}_0,$$
$$\hat{c}_1 = \frac{\hat{a}}{P} + \hat{e}_1$$

with $\|\hat{e}_0\|_\infty \leq k/2$ and $\|\hat{e}_1\|_\infty \leq k/2$. Finally, Step 6 returns the RNS representation of $(a, b) \in R^2_{n,Q_i}$ which satisfies

$$\begin{aligned}
b + as &\equiv c_0 + \hat{c}_0 + (c_1 + \hat{c}_1)s \mod (\phi(x), Q_i) \\
&\equiv c_0 + c_1 s + P^{-1}(\hat{b} + \hat{a}s) + \hat{e}_0 + \hat{e}_1 s \mod (\phi(x), Q_i) \\
&\equiv c_0 + c_1 s + P^{-1}(c_2 P s^2 + \hat{e} + \omega P Q_i) + \hat{e}_0 + \hat{e}_1 s \mod (\phi(x), Q_i) \\
&\equiv c_0 + c_1 s + c_2 s^2 + P^{-1}\hat{e} + \omega Q_i + \hat{e}_0 + \hat{e}_1 s \mod (\phi(x), Q_i) \\
&\equiv D_{Q_i}[m_0 m_1]_{\phi(x),t} + e' + P^{-1}\hat{e} + \hat{e}_0 + \hat{e}_1 s \mod (\phi(x), Q_i) \\
&\equiv D_{Q_i}[m_0 m_1]_{\phi(x),t} + e^* \mod (\phi(x), Q_i)
\end{aligned}$$

for $e^* = e' + P^{-1}\hat{e} + \hat{e}_0 + \hat{e}_1 s$. We now turn to the noise term e^*. If $P \geq 6Q_i$, $\delta_R \geq 16$, and $k > i$, then

$$\begin{aligned}
\|e^*\|_\infty &\leq \|e'\|_\infty + P^{-1}\|\hat{e}\|_\infty + \left\|\hat{d}_0 s\right\|_\infty + \left\|\hat{d}_1\right\|_\infty \\
&\leq \|e'\|_\infty + \frac{Q_i(i+1)}{2P}\delta_R^2 \|s\|_\infty + \frac{k}{2} + \frac{k\delta_R\|s\|_\infty}{2} \\
&\leq \|e'\|_\infty + \delta_R^2 k(\frac{1}{12} + \frac{1}{\delta_R^2} + \frac{1}{2\delta_R}) \\
&\leq E + \frac{1}{8}\delta_R^2 k.
\end{aligned}$$

Proof of Lemma 4.8. By assumption the input of Algorithm 24 is the RNS representation in basis $\mathcal{B}$ of some $(c_0, c_1, c_2) \in R^3_{n,Q_i}$ satisfying

$$c_0 + c_1 s + c_2 s^2 \equiv [m_0 m_1]_{\phi(x),t} + te' \mod (\phi(x), Q_i).$$

for $\|e'\|_\infty \leq E$. From Step 1, $(\tilde{c}_2^{(0)}, \ldots, \tilde{c}_2^{(k-1)}, c_2^{(0)}, \ldots, c_2^{(i)})$ is the RNS representation of $\tilde{c}_2 \in R_{n,PQ_i}$ in basis $\mathcal{D}$ satisfying $\tilde{c}_2 = c_2 + Q_i e$ for some $e \in R_n$ with $\|\tilde{c}_2\|_\infty \leq Q_i(i+$

1)/2 by Lemma 4.4. After Step 3, note $(\hat{a}^{(j)}, \hat{b}^{(j)})_{0\le j\le i+k}$ is the RNS representation of some $(\hat{a}, \hat{b}) \in R^2_{n,PQ_i}$ satisfying

$$\begin{aligned}
\hat{b} + \hat{a}s &\equiv \tilde{c}_2\tilde{\mathtt{k}}_1 + \tilde{c}_2\tilde{\mathtt{k}}_0 s \mod (\phi(x), PQ_i) \\
&\equiv \tilde{c}_2(-\tilde{\mathtt{k}}_0 s + Ps^2 + t\tilde{e}) + \tilde{c}_2\tilde{\mathtt{k}}_0 s \mod (\phi(x), PQ_i) \\
&\equiv (c_2 + Q_i e)(-\tilde{\mathtt{k}}_0 s + Ps^2 + t\tilde{e}) + (c_2 + Q_i e)\tilde{\mathtt{k}}_0 s \mod (\phi(x), PQ_i) \\
&\equiv (c_2 + Q_i e)(Ps^2 + t\tilde{e}) \mod (\phi(x), PQ_i) \\
&\equiv c_2 Ps^2 + tc_2\tilde{e} + PQ_i e s^2 + tQ_i e\tilde{e} \mod (\phi(x), PQ_i) \\
&\equiv c_2 Ps^2 + t(c_2\tilde{e} + Q_i e\tilde{e}) \mod (\phi(x), PQ_i) \\
&\equiv c_2 Ps^2 + t\hat{e} \mod (\phi(x), PQ_i)
\end{aligned}$$

for $\hat{e} = c_2\tilde{e} + Q_i e\tilde{e} = \tilde{c}_2\tilde{e}$. Note that as $\tilde{e} \leftarrow \chi_\rho$, we have $\|\hat{e}\|_\infty = \|\tilde{c}_2\tilde{e}\|_\infty \le Q_i(i+1)\rho^2/2$. Furthermore, there exists $\omega \in R_n$ such that

$$\hat{b} + \hat{a}s \equiv c_2 Ps^2 + t\hat{e} + \omega PQ_i \mod \phi(x).$$

By Lemma 4.6, Step 4 returns the RNS representation of some $\hat{c}_0 \in R_{n,Q_i}$ and $\hat{c}_1 \in R_{n,Q_i}$ satisfying

$$\begin{aligned}
\hat{c}_0 &= \frac{\hat{b}}{P} + t \cdot \hat{e}_0, \\
\hat{c}_1 &= \frac{\hat{a}}{P} + t \cdot \hat{e}_1
\end{aligned}$$

with $\|\hat{e}_0\|_\infty \le k/2$ and $\|\hat{e}_1\|_\infty \le k/2$. Finally, Step 6 returns the RNS representation of $(a, b) \in R^2_{n,Q_i}$ which satisfies

$$\begin{aligned}
b + as &\equiv c_0 + \hat{c}_0 + (c_1 + \hat{c}_1)s \mod (\phi(x), Q_i) \\
&\equiv c_0 + c_1 s + P^{-1}(\hat{b} + \hat{a}s) + t\hat{e}_0 + t\hat{e}_1 s \mod (\phi(x), Q_i) \\
&\equiv c_0 + c_1 s + P^{-1}(c_2 Ps^2 + t\hat{e} + \omega PQ_i) + t\hat{e}_0 + t\hat{e}_1 s \mod (\phi(x), Q_i) \\
&\equiv c_0 + c_1 s + c_2 s^2 + P^{-1}t\hat{e} + \omega Q_i + t\hat{e}_0 + t\hat{e}_1 s \mod (\phi(x), Q_i) \\
&\equiv [m_0 m_1]_{\phi(x),t} + te' + P^{-1}t\hat{e} + t\hat{e}_0 + t\hat{e}_1 s \mod (\phi(x), Q_i) \\
&\equiv [m_0 m_1]_{\phi(x),t} + te^* \mod (\phi(x), Q_i)
\end{aligned}$$

for $e^* = e' + P^{-1}\hat{e} + \hat{e}_0 + \hat{e}_1 s$. The bound on e^* follows identically as in the proof of Lemma 4.7.

Proof of Lemma 4.9. By assumption the input of Algorithm 24 is the RNS representation in basis $\mathcal{B}$ of some $(c_0, c_1, c_2) \in R^3_{n,Q_i}$ satisfying

$$c_0 + c_1 s + c_2 s^2 \equiv m_0 m_1 + e' \mod (\phi(x), Q_i).$$

for $\left\|e'\right\|_\infty \le E$. By an identical argument to the proof of Lemma 4.7, the output of Algorithm 24 is the RNS representation in basis $\mathcal{B}$ of $(a, b) \in R_{n,Q_i}$ satisfying

$$b + as \equiv m_0 m_1 + e^*$$

with $\|e^*\|_\infty \le E + \frac{1}{8}\delta_R^2 k$.