# A Full RNS Variant of Approximate Homomorphic Encryption

Jung Hee Cheon[1], Kyoohyung Han[1], Andrey Kim[1],
Miran Kim[2], and Yongsoo Song[3]

[1] Seoul National University
{jhcheon, satanigh, kimandrik}@snu.ac.kr
[2] University of Texas, Health Science Center at Houston
miran.kim@uth.tmc.edu
[3] University of California, San Diego
yongsoosong@ucsd.edu

**Abstract.** The technology of Homomorphic Encryption (HE) has improved rapidly in a few years. The newest HE libraries are efficient enough to use in practical applications. For example, Cheon et al. (ASIACRYPT'17) proposed an HE scheme with support for arithmetic of approximate numbers. An implementation of this scheme shows the best performance in computation over the real numbers. However, its implementation could not employ a core optimization technique based on the Residue Number System (RNS) decomposition and the Number Theoretic Transformation (NTT).

In this paper, we present a variant of approximate homomorphic encryption which is optimal for implementation on standard computer system. We first introduce a new structure of ciphertext modulus which allows us to use both the RNS decomposition of cyclotomic polynomials and the NTT conversion on each of the RNS components. We also suggest new approximate modulus switching procedures without any RNS composition. Compared to previous exact algorithms requiring multi-precision arithmetic, our algorithms can be performed by using only word size (64-bit) operations.

Our scheme achieves a significant performance gain from its full RNS implementation. For example, compared to the earlier implementation, our implementation showed speed-ups 17.3, 6.4, and 8.3 times for decryption, constant multiplication, and homomorphic multiplication, respectively, when the dimension of a cyclotomic ring is 32768. We also give experimental result for evaluations of some advanced circuits used in machine learning or statistical analysis. Finally, we demonstrate the practicability of our library by applying to machine learning algorithm. For example, our single core implementation takes 1.8 minutes to build a logistic regression model from encrypted data when the dataset consists of 575 samples, compared to the previous best result 3.5 minutes using four cores.

**Keywords.** Homomorphic encryption, approximate arithmetic, residue number system.

## 1 Introduction

As the growth of big data analysis have led to many concerns about security and privacy of data, researches on secure computation have been highlighted in cryptographic community. Homomorphic Encryption (HE) is a cryptosystem that allows an arbitrary circuit to be evaluated on encrypted data without decryption. It has been one of the most promising solutions that make it possible to outsource computation and securely aggregate sensitive information of individuals. After the first construction of fully homomorphic encryption by Gentry [20], several researches [16, 7, 18, 17, 11] have improved the efficiency of HE schemes.

There are a few software implementations of HE schemes based on the Ring Learning with Errors (RLWE) problem such as `HElib` [25] of the BGV scheme [7] and `SEAL` [32] of the BFV scheme [6, 18]. These HE schemes are constructed over the residue ring of a cyclotomic ring (with a huge characteristic) so they manipulate modulo operations between high-degree polynomials, resulting in a performance degradation. For an efficient implementation of polynomial arithmetic, Gentry et al. [21] suggested a representation of cyclotomic polynomials, called the *double-CRT* representation, based on the Chinese Remainder Theorem (CRT). The first CRT layer uses the Residue Number System (RNS) in order to decompose a polynomial into a tuple of polynomials with smaller moduli. The second layer converts each of small polynomials into a vector of modulo integers via the Number Theoretic Transform (NTT). In the double-CRT representation, an arbitrary polynomial is identified with a matrix consisting of small integers, and this enables an efficient polynomial arithmetic by performing componentwise modulo operations. This technique became one of the core optimization techniques used in the implementations of HE schemes [25, 32, 1].

Cheon et al. [11] recently suggested an HE scheme for arithmetic of approximate numbers, called `HEAAN`. The main idea of their construction is to consider an RLWE error as a part of an error occurring during approximate computations. Besides homomorphic addition and multiplication, it supports an approximate rounding operation of significant digits on packed ciphertexts. This approximate HE scheme shows remarkable performance in real-world applications that require arithmetic over the real numbers [28, 27].

However, the original scheme had one significant problem in the use of the double-CRT representation. The rounding operation of `HEAAN` can be done by dividing an encrypted plaintext by a ratio of two consecutive ciphertext moduli, so a ciphertext modulus should be chosen as a power of two (or some prime). This parameter choice makes it hard to implement the HE scheme on the RNS representation. Consequently, the previous implementation [10] took a longer time to perform homomorphic operations than other implementations of HE schemes under the same parameter setting.

**Our Contribution.** In this paper, we present a variant of `HEAAN` based on the double-CRT representation of cyclotomic polynomial ring elements. The main idea is to exploit a basis consisting of some approximate values of a fixed base as our moduli chain. Every encrypted message in `HEAAN` contains a small noise

from approximate computations. The approximate rounding operation of our scheme yields an additional error from approximation, but it does not destroy the significant digits of an encrypted message as long as the precision of the approximate bases is higher than the precision of the plaintexts. In addition, by selecting approximate bases satisfying some condition for the NTT conversion, we take the advantages of double-CRT representation while maintaining the functionalities of the original scheme.

We also introduce some modulus switching algorithms that can be computed without RNS composition. To be more precise, some homomorphic operations of the original `HEAAN` scheme (e.g. homomorphic multiplication) require *non-arithmetic* operations such as modulus raising and reduction, which are difficult to perform based on the RNS representation. As a result, the previous implementation required multi-precision arithmetics instead of working on typical word-size integers in hardware architecture (e.g. 64-bit processor). Our new modulus switching techniques can substitute the non-arithmetic operations in the previous scheme. These algorithms are RNS-friendly, that is, they can be represented using only word operations without RNS composition.

We implemented our scheme and compared with the original one to show the performance benefit from a full RNS system. For efficient implementation in the NTT and modulus operations, we adapt harvey's butterfly and barrett modulus reduction techniques. Our full RNS variant improves the performance of basic operations by nearly ten times compared to the original `HEAAN` [11, 10]. The decryption and homomorphic multiplication timings are reduced from 135 and 1,355 milliseconds down to 7.8 and 164 milliseconds, respectively, when evaluating a circuit of depth 10.

We also present experimental results for homomorphic evaluation of analytic functions and statistic functions. It took 160 milliseconds to compute the multiplicative inverse, exponential function, or sigmoid function with inputs of 32-bit precision on $2^{13}$ slots, yielding an amortized time of 20 microseconds per slot. In the case of statistic functions, it took 307 and 518 milliseconds to obtain the mean and variance of $2^{13}$ real numbers, respectively.

Finally, we implemented a variant of the gradient descent algorithm to show that our HE library can perform complex computations in real-world applications. Our single-core implementation took about 1.8 minutes to obtain a logistic regression model from homomorphically encrypted dataset consisting of 575 samples each of which has eight features and a binary class information, compared to previous best result of 3.5 minutes using a machine with four cores [27].

**Technical Details.** Let $N$ be a power-of-two integer and $R = \mathbb{Z}[X]/(X^N+1)$ be the ring of integers of the $(2N)$-th cyclotomic field. For a fixed base $q$, we choose an RNS basis $\{q_0, \ldots, q_L\}$ which is a set of coprime integers of approximately the same size as the base $q$. For an integer $0 \leq \ell \leq L$, a ciphertext at level-$\ell$ is a pair of polynomials in $R_{Q_\ell} = R/(Q_\ell \cdot R)$ for $Q_\ell = \prod_{i=0}^{\ell} q_i$. The rescaling procedure transforms a level $\ell$ encryption of $m$ into a level $(\ell-1)$ encryption of $q_\ell^{-1} \cdot m$, which is an approximation of $q^{-1} \cdot m$ with almost the same precision. The original scheme is more flexible in choice of ciphertext modulus since it can

rescale a plaintext by an arbitrary number compared to the fixed base $q$ of our RNS variant. However, our scheme has a significant improvement in performance.

Our scheme can support the NTT representation of RNS decomposed polynomials as the double-CRT representation in the BGV scheme [7, 21]. The NTT conversion can be done efficiently when the approximate bases $q_\ell$'s are prime numbers satisfying $q_\ell \equiv 1 \pmod{2N}$. We give a list of candidate bases to show that there are sufficiently many distinct primes satisfying both conditions for the double-CRT representation.

The homomorphic multiplication algorithm of `HEAAN` includes modulus switching procedures that convert an element of $R_Q$ into $R_{P \cdot Q}$ for a sufficiently large integer $P$ and switch back to the original modulus $Q$. These non-arithmetic operations are difficult to perform on the RNS system, so one should recover the coefficient representation of an input polynomial. For an optimization, we adapt an idea of Barjard et al. [3] to suggest approximate modulus switching algorithms with small errors. Instead of exact computation in the original scheme, our approximate modulus raising algorithm finds an element $\tilde{a} \in R_{P \cdot Q}$ satisfying $\tilde{a} \equiv a \pmod{Q}$ and $\|\tilde{a}\| \ll P \cdot Q$ for a given polynomial $a \in R_Q$. Conversely, the approximate modulus reduction algorithm returns an element $b \in R_Q$ such that $P \cdot b \approx \tilde{b}$ for an input polynomial $\tilde{b} \in R_{P \cdot Q}$. These procedures give relaxed conditions on output polynomials, but we can construct algorithms that can be performed on the RNS representation. In addition, we show that the correctness of the HE system is still guaranteed with some small additional error.

**Related Works.** There have been several studies [15, 14, 5, 8] about homomorphic arithmetic over real or integral numbers besides the `HEAAN` scheme. However, these approaches do not support the rounding operation which is a core algorithm in approximate computation, and consequently, the required bit-size of a ciphertext modulus grows exponentially on the depth of a circuit to be evaluated.

Many of HE schemes use a polynomial ring structure with large coefficients. Some recent researches accelerated expensive ring operations by exploiting the RNS representation. Bajard et al. [3] proposed a full RNS variant of the BFV scheme [6, 18]. Their implementation could avoid the need of conversion between RNS and coefficient representations of an ring element during homomorphic computations. After that, Halevi et al. [24] presented a simplified method with reduced noise growth. Based on this idea, one can implement an HE scheme without any numerical library for big integer arithmetics. This technique has been applied to `SEAL` [32] after v2.3.1.

**Road-map.** In Section 2, we review the basics of the `HEAAN` scheme and introduce fast base conversion. In Section 3, we present a method to improve overall homomorphic operations from RNS representation. In Section 4, we describe a full RNS variant of `HEAAN`. Finally, Section 5 shows experimental results with optimization techniques.

## 2 Background

All logarithms are base 2 unless otherwise indicated. We denote vectors in bold, e.g. $\mathbf{a}$, and every vector in this paper will be a column vector. We denote by $\langle \cdot, \cdot \rangle$ the usual dot product of two vectors. For a real number $r$, $\lfloor r \rceil$ denotes the nearest integer to $r$, rounding upwards in case of a tie. For an integer $q$, we identify $\mathbb{Z} \cap (-q/2, q/2]$ as a representative of $\mathbb{Z}_q$ and use $[a]_q$ to denote the reduction of the integer $a$ modulo $q$ into that interval. We use $x \leftarrow D$ to denote the sampling $x$ according to a distribution $D$ and $U(S)$ denotes the uniform distribution over $S$ when $S$ is a finite set. We let $\lambda$ denote the security parameter throughout the paper: all known valid attacks against the cryptographic scheme under scope should take $\Omega(2^\lambda)$ bit operations. A finite ordered set $\mathcal{B} = \{p_0, p_1, \ldots, p_{k-1}\}$ of integers is called a *basis* if it is pairwise coprime.

### 2.1 Approximate Homomorphic Encryption

Cheon et al. [11] proposed an HE scheme that supports an approximate arithmetic on encrypted data. The main idea is to consider an error of homomorphic operation (e.g. encryption, multiplication) as part of computational error in approximate computation.

For a power-of-two integer $N$, we denote by $K = \mathbb{Q}[X]/(X^N+1)$ the $(2N)$-th cyclotomic field and $R = \mathbb{Z}[X]/(X^N + 1)$ its ring of integers. The residue ring modulo an integer $q$ is denoted by $R_q = R/qR$. The `HEAAN` scheme uses a fixed *base* integer $q$ and constructs a chain of moduli $Q_\ell = q^\ell$ for $1 \le \ell \le L$. For a polynomial $m(X) \in K$, a ciphertext $\mathsf{ct}$ is called an encryption of $m(X)$ at level $\ell$ if $\mathsf{ct} \in R_{Q_\ell}^2$ and $[\langle \mathsf{ct}, \mathsf{sk} \rangle]_{Q_\ell} \approx m(X)$. Homomorphic operations between ciphertexts of `HEAAN` can be done by the key-switching with special modulus suggested in [21]. For input encryptions of $m_1(X)$ and $m_2(X)$ at a level $\ell$, their homomorphic addition and multiplication satisfy $[\langle \mathsf{ct}_{\mathsf{add}}, \mathsf{sk} \rangle]_{Q_\ell} \approx m_1(X) + m_2(X)$ and $[\langle \mathsf{ct}_{\mathsf{mult}}, \mathsf{sk} \rangle]_{Q_\ell} \approx m_1(X) \cdot m_2(X)$, respectively.

The main advantage of this scheme comes from its intrinsic operation called the rescaling procedure. The rescaling algorithm, denoted by $\mathtt{RS}(\cdot)$, transforms a level $\ell$ encryption of $m(X)$ into an encryption of $q^{-1} \cdot m(X)$ at level $(\ell - 1)$. It can be considered as an approximate rounding operation or an approximate extraction of the most significant bits of the encrypted plaintext. By reducing the size of the plaintext, we can reduce the speed of modulus consumption in the following computation.

For packing of multiple messages, there has been suggested a method to identify an element of a cyclotomic field with a complex vector via a variant of the canonical embedding. Let $\zeta = \exp(-\pi i/N)$ be a $(2N)$-th root of unity in $\mathbb{C}$. Recall that the canonical embedding of $K$ is defined by $a(X) \mapsto (a(\zeta), a(\zeta^3), \ldots, a(\zeta^{2N-1}))$. Note that there is no need to store all entries of $\sigma(a)$ to recover $a(X)$ since $a(\zeta^j) = \overline{a(\zeta^{2N-j})}$. We denote by $\tau : K \to \mathbb{C}^{N/2}$ a variant of the canonical embedding, defined by

$$\tau : a(X) \mapsto (a(\zeta), a(\zeta^5), \ldots, a(\zeta^{2N-3}))_{0 \le j < N/2},$$

and use it as a decoding function for HEAAN. The inverse of this homomorphism $\tau$ is used as the encoding function to pack $(N/2)$ complex numbers in a single polynomial.

This HE scheme can be applied to fixed-point arithmetic on real (complex) numbers. We multiply a scale factor of $q$ to a number $z$ with finite precision and use the value $m = q \cdot z$ for encryption. Then encryption of $m$ will satisfy $[\langle \mathsf{ct}, \mathsf{sk} \rangle]_{Q_\ell} \approx q \cdot z$, which is an approximate and a scaled value of $z$. A product of two encryptions of $q \cdot z_1$ and $q \cdot z_2$ will return an encryption of $q^2 \cdot z_1 z_2$, which is a scaled value of $z_1 \cdot z_2$ by $q^2$. We can perform the rescaling procedure to maintain the original scaling factor $q$.

### 2.2 The RNS Representation

Let $\mathcal{B} = \{p_0, \ldots, p_{k-1}\}$ be a basis and let $P = \prod_{i=0}^{k-1} p_i$. We denote by $[\cdot]_{\mathcal{B}}$ the map from $\mathbb{Z}_P$ to $\prod_{i=0}^{k-1} \mathbb{Z}_{p_i}$, defined by $a \mapsto [a]_{\mathcal{B}} = ([a]_{p_i})_{0 \le i < k}$. It is a ring isomorphism from the Chinese Remainder Theorem (CRT) and $[a]_{\mathcal{B}}$ is called the *residue number system* (RNS) representation of $a \in \mathbb{Z}_P$. The main advantage of the RNS representation is to perform component-wise arithmetic operations in the small rings $\mathbb{Z}_{p_i}$, which reduces the asymptotic and practical computation cost. This ring isomorphism over the integers can be naturally extended to a ring isomorphism $[\cdot]_{\mathcal{B}} : R_P \to R_{p_0} \times \cdots \times R_{p_{k-1}}$ by applying it coefficient-wise over the cyclotomic rings.

### 2.3 Fast Basis Conversion

Brakerski [6] introduced a scale-invariant HE scheme based on the LWE problem, and Fan and Vercauteren [18] suggested its ring-based variant called BFV. Barjard et al. [3] proposed a variant of the BFV scheme that maintains the RNS representation of ciphertexts during homomorphic computation. This scheme presents a new algorithm, called the fast basis conversion, to convert the residue of a polynomial into a new basis that is coprime to the original basis.

More precisely, for a basis $\{p_0, \ldots, p_{k-1}, q_0, \ldots, q_{\ell-1}\}$, let $\mathcal{B} = \{p_0, \ldots, p_{k-1}\}$ and $\mathcal{C} = \{q_0, \ldots, q_{\ell-1}\}$ be its subbases. Let us denote their products by $P = \prod_{i=0}^{k-1} p_i$ and $Q = \prod_{j=0}^{\ell-1} q_j$, respectively. Then one can convert the RNS representation $[a]_{\mathcal{C}} = (a^{(0)}, \ldots, a^{(\ell-1)}) \in \mathbb{Z}_{q_0} \times \cdots \times \mathbb{Z}_{q_{\ell-1}}$ of an integer $a \in \mathbb{Z}_Q$ into an element of $\mathbb{Z}_{p_0} \times \cdots \times \mathbb{Z}_{p_{k-1}}$ by computing

$$\mathtt{Conv}_{\mathcal{C} \to \mathcal{B}}([a]_{\mathcal{C}}) = \left( \sum_{j=0}^{\ell-1} [a^{(j)} \cdot \hat{q}_j^{-1}]_{q_j} \cdot \hat{q}_j \pmod{p_i} \right)_{0 \le i < k},$$

where $\hat{q}_j = \prod_{j' \neq j} q_{j'} \in \mathbb{Z}$. We note that $\sum_{j=0}^{\ell-1} [a^{(j)} \cdot \hat{q}_j^{-1}]_{q_j} \cdot \hat{q}_j = a + Q \cdot e$ for some small $e \in \mathbb{Z}$ satisfying $|a + Q \cdot e| \le (\ell/2) \cdot Q$. This implies that $\mathtt{Conv}_{\mathcal{C} \to \mathcal{B}}([a]_{\mathcal{C}}) = [a + Q \cdot e]_{\mathcal{B}}$ can be considered as the RNS representation of the integer $a + Q \cdot e$ with respect to the basis $\mathcal{B}$.

## 3 Approximate Bases and Full RNS Modulus Switching

The approximate HE scheme of Cheon et al. [11] has its own advantages in arithmetic of approximate numbers. However, a ciphertext modulus could not be chosen as a product of coprime integers, so its implementation [10] requires expensive multi-precision modular arithmetic. In this section, we introduce an idea to avoid the use of a power-of-two base ciphertext modulus and enable the RNS decomposition in the `HEAAN` scheme. We also propose new algorithms to switch a ciphertext modulus on the RNS components.

### 3.1 Approximate Basis

The main advantage of `HEAAN` comes from the rescaling algorithm $\mathtt{RS}(\cdot)$. It allows us to perform the rounding of an encrypted plaintext, that is, we can efficiently convert an encryption of $m$ into a ciphertext encrypting the scaled message $q^{-1} \cdot m$. In the case of its application to fixed-point arithmetic, for example, we multiply fixed-point numbers $z_i$ by a common scale factor of $q$ to maintain the precision of plaintexts. After homomorphic multiplication, we obtain an encryption of the product $q^2 \cdot z_1 z_2$ of two numbers $q \cdot z_1$ and $q \cdot z_2$. Then we perform the rescaling algorithm to get an encryption of $q \cdot z_1 z_2$ and maintain the original scale factor $q$. For this reason, the ciphertext modulus should be chosen as a power of a fixed base $Q_\ell = q^\ell$ to have the same scaling ratio. This point made it difficult to use the RNS representation on `HEAAN`.

To overcome this obstacle, we propose an idea to use an RNS basis consisting of *approximate* values of a fixed base. In more detail, given the scale factor $q$ and bit precision $\eta$, we find a basis $\mathcal{C} = \{q_0, \ldots, q_L\}$ such that $q/q_\ell \in (1-2^{-\eta}, 1+2^{-\eta})$ for $\ell = 1, \ldots, L$. This approximate basis allows us to use the RNS representation of polynomials while keeping the functionality of the HE scheme. We set the level $\ell$ ciphertext modulus as $Q_\ell = \prod_{i=0}^{\ell} q_i$, so that the ciphertext moduli in the consecutive levels have almost the same ratio $Q_\ell / Q_{\ell-1} = q_\ell \approx q$. The rescaling algorithm with a factor of $q_\ell$ converts an encryption of $m$ at level $\ell$ into an encryption of $q_\ell^{-1} \cdot m$ at level $(\ell - 1)$. This operation has an additional error from the approximation of $q$, but we can manage the size of an error not to destroy the significant digits of a plaintext. An approximation error is bounded by

$$\left| q_\ell^{-1} \cdot m - q^{-1} \cdot m \right| = \left| 1 - q_\ell^{-1} \cdot q \right| \cdot \left| q^{-1} \cdot m \right| \leq 2^{-\eta} \cdot \left| q^{-1} \cdot m \right|,$$

so it does not destroy the significant digits of an encrypted plaintext when $\eta$ is sufficiently larger than the bit precision of an encrypted plaintext.

### 3.2 Approximate Modulus Switching

The use of an approximate basis enables an implementation of the `HEAAN` scheme using the RNS representation. However, `HEAAN` includes some non-arithmetic operations that cannot be directly implemented on the RNS components. Specifically, homomorphic multiplication and rescaling procedure require an exact

**Algorithm 1** Approximate Modulus Raising

1: **procedure** $\texttt{ModUp}_{\mathcal{C}\to\mathcal{D}}(a^{(0)}, a^{(1)}, \ldots, a^{(\ell-1)})$
2: $\quad(\tilde{a}^{(0)}, \ldots, \tilde{a}^{(k-1)}) \leftarrow \texttt{Conv}_{\mathcal{C}\to\mathcal{B}}([a]_{\mathcal{C}})$.
3: **return** $(\tilde{a}^{(0)}, \ldots, \tilde{a}^{(k-1)}, a^{(0)}, \ldots, a^{(\ell-1)})$.
4: **end procedure**

modulus switching algorithm, and the key-switching technique for rotation and conjugation also contains the same operation (see [11, 9] for details).

We remark that the goal of modulus switching algorithms in [11] can be reduced to a problem that finds a ciphertext with a small error while keeping the correctness of the HE scheme. In this section, we propose an idea to approximately perform the modulus switching algorithms on the RNS representation. A full RNS variant of `HEAAN` will be described in the next section based on this method. Throughout this paper, we will denote by $\mathcal{D} = \{p_0, \ldots, p_{k-1}, q_0, \ldots, q_{\ell-1}\}$, $\mathcal{B} = \{p_0, \ldots, p_{k-1}\}$, and $\mathcal{C} = \{q_0, \ldots, q_{\ell-1}\}$ an RNS basis and its subbases, respectively, with $P = \prod_{i=0}^{k-1} p_i$ and $Q = \prod_{j=0}^{\ell-1} q_j$.

**Approximate Modulus Raising.** Suppose that we are given the RNS representation $[a]_{\mathcal{C}}$ of an integer $a \in \mathbb{Z}_Q$. The purpose of the approximate modulus raising algorithm, denoted by `ModUp`, is to find the RNS representation of an integer $\tilde{a} \in \mathbb{Z}_{PQ}$ with respect to the basis $\mathcal{D}$ satisfying two conditions $\tilde{a} \equiv a \pmod{Q}$ and $|\tilde{a}| \ll P \cdot Q$. From the first condition $[\tilde{a}]_{\mathcal{C}} = [a]_{\mathcal{C}}$, we only need to generate the RNS representation of $\tilde{a}$ with the basis $\mathcal{B}$ and it can be done by applying the fast conversion algorithm. See Algorithm 1 for a description of the approximate modulus raising.

As described in Section 2.3, the fast conversion algorithm in Algorithm 1 returns $[a + Q \cdot e]_{\mathcal{B}} \in \prod_{i=0}^{k-1} \mathbb{Z}_{p_i}$ for some integer $e$ with $|e| \leq \ell/2$ . Therefore, the output of `ModUp` algorithm is the RNS representation of $\tilde{a} := a + Q \cdot e$ with respect to the basis $\mathcal{D} = \mathcal{B} \cup \mathcal{C}$, as desired.

**Approximate Modulus Reduction.** Contrary to the modulus raising algorithm, the approximate modulus reduction algorithm, denoted by `ModDown`, takes an RNS representation $[\tilde{b}]_{\mathcal{D}}$ of an integer $\tilde{b} \in \mathbb{Z}_{P \cdot Q}$ as an input and aims to compute $[b]_{\mathcal{C}}$ for some integer $b \in \mathbb{Z}_Q$ satisfying $b \approx P^{-1} \cdot \tilde{b}$.

We point out that the goal of approximate modulus reduction is reduced to a problem of finding small $\tilde{a} = \tilde{b} - P \cdot b$ satisfying $\tilde{a} \equiv \tilde{b} \pmod{P}$. The RNS representation $[\tilde{b}]_{\mathcal{D}}$ is the concatenation of $[\tilde{b}]_{\mathcal{B}}$ and $[\tilde{b}]_{\mathcal{C}}$. We first take the first component $[\tilde{b}]_{\mathcal{B}} = (\tilde{b}^{(0)}, \ldots, \tilde{b}^{(k-1)})$, which is the same as $[a]_{\mathcal{B}}$ for $a = [\tilde{b}]_P \in \mathbb{Z}_P$. Then we apply the fast conversion algorithm to compute the RNS representation $[\tilde{a}]_{\mathcal{C}}$ of $\tilde{a} = a + P \cdot e$ for some small $e$. Note that $\tilde{a} \equiv \tilde{b} \pmod{P}$ and $|\tilde{a}| \ll P \cdot Q$ from the property of $\texttt{Conv}_{\mathcal{B}\to\mathcal{C}}(\cdot)$. Finally, we derive the RNS representation of $b = P^{-1} \cdot (\tilde{b} - \tilde{a})$ with respect to the basis $\mathcal{C}$ by computing $\left(\prod_{i=0}^{k-1} p_i\right)^{-1} \cdot \left([\tilde{b}]_{\mathcal{C}} - [\tilde{a}]_{\mathcal{C}}\right) \in \prod_{j=0}^{\ell-1} \mathbb{Z}_{q_j}$. See Algorithm 2 for a description.

**Algorithm 2** Approximate Modulus Reduction

**procedure** $\texttt{ModDown}_{\mathcal{D}\to\mathcal{C}}(\tilde{b}^{(0)}, \tilde{b}^{(1)}, \ldots, \tilde{b}^{(k+\ell-1)})$
  $(\tilde{a}^{(0)}, \ldots, \tilde{a}^{(\ell-1)}) \leftarrow \texttt{Conv}_{\mathcal{B}\to\mathcal{C}}(\tilde{b}^{(0)}, \ldots, \tilde{b}^{(k-1)})$
  **for** $0 \le j < \ell$ **do**
    $b^{(j)} = \left(\prod_{i=0}^{k-1} p_i\right)^{-1} \cdot (\tilde{b}^{(k+j)} - \tilde{a}^{(j)}) \pmod{q_j}$.
  **end for**
**return** $(b^{(0)}, \ldots, b^{(\ell-1)})$.
**end procedure**

**Word Operations.** In the rest of the paper, the arithmetic operations (e.g. addition and multiplication) modulo a "word-size" integer will be called the *word operations*. Now suppose that $p_i$'s and $q_j$'s are word-size integers. As mentioned before, the fast conversion algorithm $\texttt{Conv}_{\mathcal{C}\to\mathcal{B}}([a]_{\mathcal{C}})$ outputs the tuple $\left(\sum_{j=0}^{\ell-1}[a^{(j)} \cdot \hat{q}_j^{-1}]_{q_j} \cdot \hat{q}_j \pmod{p_i}\right)_{0\le i<k}$ for $\hat{q}_j = \prod_{j'\neq j} q_{j'}$. Each component can be computed using the values $[\hat{q}_j^{-1}]_{q_j} = \prod_{j'\neq j} q_{j'}^{-1} \pmod{q_j}$ and $[\hat{q}_j]_{p_i} = \prod_{j'\neq j} q_{j'} \pmod{p_i}$ while avoiding the computation of *big* integers $\hat{q}_j$. In addition, if we pre-compute and store these values, which depend only on the bases $\mathcal{B}$ and $\mathcal{C}$, then the computation cost of $\texttt{Conv}_{\mathcal{C}\to\mathcal{B}}(\cdot)$ algorithm can be reduced down to $O(k \cdot \ell)$ word operations.

**Complexity of Approximate Modulus Switching.** Our modulus switching algorithms have an advantage, in that they can be computed only using word operations. For example, $\texttt{ModUp}_{\mathcal{C}\to\mathcal{D}}([a]_{\mathcal{C}})$ requires exactly the same computation as $\texttt{Conv}_{\mathcal{C}\to\mathcal{B}}([a]_{\mathcal{C}})$, so its total complexity is bounded by $O(k\cdot\ell)$ word operations. The approximate modulus reduction algorithm needs to compute $b^{(j)} = P^{-1} \cdot (\tilde{b}^{(k+j)} - \tilde{a}^{(j)}) \pmod{q_j}$ for $0 \le j < \ell$ as well as the fast conversion algorithm. The computation of $b^{(j)}$'s can be done in $O(\ell)$ word operations using the pre-computable constants $[P^{-1}]_{q_j} = \left(\prod_{i=0}^{k-1} p_i\right)^{-1} \pmod{q_j}$. Therefore, the total complexity of $\texttt{ModDown}$ is bounded by $O(k \cdot \ell + \ell) = O(k \cdot \ell)$ word operations.

The approximate modulus switching algorithms can be extended to algorithms over the polynomial rings as

$$\texttt{ModUp}_{\mathcal{C}\to\mathcal{D}}(\cdot) : \prod_{j=0}^{\ell-1} R_{q_j} \to \prod_{i=0}^{k-1} R_{p_i} \times \prod_{j=0}^{\ell-1} R_{q_j},$$
$$\texttt{ModDown}_{\mathcal{D}\to\mathcal{C}}(\cdot) : \prod_{i=0}^{k-1} R_{p_i} \times \prod_{j=0}^{\ell-1} R_{q_j} \to \prod_{j=0}^{\ell-1} R_{q_j}$$

by applying them coefficient-wise. These operations require $O(k \cdot \ell \cdot N)$ word operations where $N$ is a degree of a power-of-two cyclotomic ring.

## 4 A Full RNS Variant of the Approximate HE

In this section, we propose a variant of HEAAN based on the full RNS representation. For simplicity, we choose a power-of-two integer $N$ and consider the $(2N)$-th cyclotomic field $K = \mathbb{Q}[X]/(X^N+1)$ and its ring of integers $R = \mathbb{Z}[X]/(X^N+1)$. An arbitrary element of $K$ is expressed as a polynomial with rational coefficients of degree strictly less than $N$, and identified with the vector of its coefficients in $\mathbb{Q}^N$. The rounding operation on $K$ and the modulo operation on $R$ will be defined by the coefficient-wise rounding and modulo operations, respectively. In the following, we present a concrete description of a full RNS variant of HEAAN.

$\underline{\texttt{Setup}(q, L, \eta; 1^\lambda)}$. A base integer $q$, the number of levels $L$, and the bit precision $\eta$ are given as inputs with the security parameter $\lambda$.

- Choose a basis $\mathcal{D} = \{p_0, \ldots, p_{k-1}, q_0, q_1, \ldots, q_L\}$ such that $q_j/q \in (1 - 2^{-\eta}, 1 + 2^{-\eta})$ for $1 \le j \le L$. We write $\mathcal{B} = \{p_0, \ldots, p_{k-1}\}$, $\mathcal{C}_\ell = \{q_0, \ldots, q_\ell\}$, and $\mathcal{D}_\ell = \mathcal{B} \cup \mathcal{C}_\ell = \{p_0, \ldots, p_{k-1}, q_0, \ldots, q_\ell\}$ for $0 \le \ell \le L$. Let $P = \prod_{i=0}^{k-1} p_i$ and $Q = \prod_{j=0}^{L} q_j$.
- Choose a power-of-two integer $N$.
- Choose a secret key distribution $\chi_{\mathsf{key}}$, an encryption key distribution $\chi_{\mathsf{enc}}$, and an error distribution $\chi_{\mathsf{err}}$ over $R$.
- Let $\hat{p}_i = \prod_{0 \le i' < k, i' \ne i} p_{i'}$ for $0 \le i < k$. Compute the constants $[\hat{p}_i]_{q_j}$ and $[\hat{p}_i^{-1}]_{p_i}$ for $0 \le i < k$, $0 \le j \le L$.
- Compute the constants $[P^{-1}]_{q_j} = \left(\prod_{i=0}^{k-1} p_i\right)^{-1} \pmod{q_j}$ for $0 \le j \le L$.
- Let $\hat{q}_{\ell,j} = \prod_{0 \le j' \le \ell, j' \ne j} q_{j'}$ for $0 \le j \le \ell \le L$. Compute the constants $[\hat{q}_{\ell,j}]_{p_i}$ and $[\hat{q}_{\ell,j}^{-1}]_{q_j}$ for $0 \le i < k$, $0 \le j \le \ell \le L$.

The constants $[\hat{p}_i]_{q_j}$ and $[\hat{p}_i^{-1}]_{p_i}$ are necessary to compute the conversion $\texttt{Conv}_{\mathcal{B} \to \mathcal{C}_\ell}(\cdot)$ in the $\texttt{ModDown}_{\mathcal{D}_\ell \to \mathcal{C}_\ell}(\cdot)$ algorithm. The constants $[P^{-1}]_{q_j}$ are also used in the algorithm. On the other hand, the constants $[\hat{q}_{\ell,j}]_{p_i}$ and $[\hat{q}_{\ell,j}^{-1}]_{q_j}$ are used to compute $\texttt{Conv}_{\mathcal{C}_\ell \to \mathcal{B}}(\cdot)$ for the $\texttt{ModUp}_{\mathcal{C}_\ell \to \mathcal{D}_\ell}(\cdot)$ algorithm.

We choose an RNS basis $\mathcal{D}$ consisting of word-size integers, so that every homomorphic arithmetic can be expressed using word operations (e.g. uint64_t). The elements of $\mathcal{B}$ are called the special primes and used in the key-switching procedure. They do not have to be close to $q$, but their product $P$ should be large enough to get a small key-switching error. The zero-level ciphertext modulus $Q_0 = q_0$ is not necessarily approximate to the base integer $q$, but it should be larger than the modulus of the encrypted plaintext for the correctness of decryption.

$\underline{\texttt{KSGen}(s_1, s_2)}$. For given secret polynomials $s_1, s_2 \in R$, sample uniform elements $(a'^{(0)}, \ldots, a'^{(k+L)}) \leftarrow U\left(\prod_{i=0}^{k-1} R_{p_i} \times \prod_{j=0}^{L} R_{q_j}\right)$ and an error $e' \leftarrow \chi_{\mathsf{err}}$. Output

the switching key $\mathsf{swk}$ as

$$\left(\mathsf{swk}^{(0)} = (b'^{(0)}, a'^{(0)}), \ldots, \mathsf{swk}^{(k+L)} = (b'^{(k+L)}, a'^{(k+L)})\right) \in \prod_{i=0}^{k-1} R_{p_i}^2 \times \prod_{j=0}^{L} R_{q_j}^2$$

where $b'^{(i)} \leftarrow -a'^{(i)} \cdot s_2 + e' \pmod{p_i}$ for $0 \leq i < k$ and $b'^{(k+j)} \leftarrow -a'^{(k+j)} \cdot s_2 + [P]_{q_j} \cdot s_1 + e' \pmod{q_j}$ for $0 \leq j \leq L$.

This procedure generates a switching key to convert a ciphertext with the secret key $s_1$ into a ciphertext encrypting the same message with the secret key $s_2$. If $a'$ is the element of $R_{P \cdot Q}$ such that $[a']_{\mathcal{D}} = (a'^{(0)}, \ldots, a'^{(k+L)})$, then the switching key $\mathsf{swk}$ can be seen as the RNS representation of $(b', a') \in R_{P \cdot Q}$ in the basis $\mathcal{D}$ for $b' = -a' \cdot s_2 + P \cdot s_1 + e' \pmod{P \cdot Q}$.

$\underline{\texttt{KeyGen}}$.

- Sample $s \leftarrow \chi_{\mathsf{key}}$ and set the secret key as $\mathsf{sk} \leftarrow (1, s)$.
- Sample $(a^{(0)}, \ldots, a^{(L)}) \leftarrow U\left(\prod_{j=0}^{L} R_{q_j}\right)$ and $e \leftarrow \chi_{\mathsf{err}}$. Set the public key as

  $$\mathsf{pk} \leftarrow \left(\mathsf{pk}^{(j)} = (b^{(j)}, a^{(j)}) \in R_{q_j}^2\right)_{0 \leq j \leq L}$$

  where $b^{(j)} \leftarrow -a^{(j)} \cdot s + e \pmod{q_j}$ for $0 \leq j \leq L$.
- Set the evaluation key as $\mathsf{evk} \leftarrow \texttt{KSGen}(s^2, s)$.

The encryption key is the RNS representation of an RLWE sample $(b = -a \cdot s + e, a) \in R_{Q_L}^2$ in the basis $\mathcal{C}_L$. The evaluation key $\mathsf{evk}$ can be used to perform the relinearization operation during homomorphic multiplication. One can generate additional public keys for more functionalities. For example, we need to publish a rotation key (resp. conjugation key) to compute the permutation (resp. conjugation) on plaintext slots as described in [11].

$\underline{\texttt{Enc}_{\mathsf{pk}}(m)}$. For $m \in R$, sample $v \leftarrow \chi_{\mathsf{enc}}$ and $e_0, e_1 \leftarrow \chi_{\mathsf{err}}$. Output the ciphertext $\mathsf{ct} = \left(\mathsf{ct}^{(j)}\right)_{0 \leq j \leq L} \in \prod_{j=0}^{L} R_{q_j}^2$ where $\mathsf{ct}^{(j)} \leftarrow v \cdot \mathsf{pk}^{(j)} + (m + e_0, e_1) \pmod{q_j}$ for $0 \leq j \leq L$.

$\underline{\texttt{Dec}_{\mathsf{sk}}(\mathsf{ct})}$. For $\mathsf{ct} = \left(\mathsf{ct}^{(j)}\right)_{0 \leq j \leq \ell}$, output $\langle \mathsf{ct}^{(0)}, \mathsf{sk} \rangle \pmod{q_0}$.

The encryption algorithm generates the RNS representation of a ciphertext $\mathsf{ct}$ satisfying $[\langle \mathsf{ct}, \mathsf{sk} \rangle]_{Q_L} \approx m$. Thus its decryption returns an approximate value of the input plaintext. The encrypted plaintext should satisfy $\|m\|_\infty \leq q_0/2$ in order to recover a correct value.

$\underline{\texttt{Add}(\mathsf{ct}, \mathsf{ct}')}$. Given two ciphertexts $\mathsf{ct} = \left(\mathsf{ct}^{(0)}, \ldots, \mathsf{ct}^{(\ell)}\right), \mathsf{ct}' = \left(\mathsf{ct}'^{(0)}, \ldots, \mathsf{ct}'^{(\ell)}\right) \in \prod_{j=0}^{\ell} R_{q_j}^2$, output a ciphertext $\mathsf{ct}_{\mathsf{add}} = \left(\mathsf{ct}_{\mathsf{add}}^{(j)}\right)_{0 \leq j \leq \ell}$ where $\mathsf{ct}_{\mathsf{add}}^{(j)} \leftarrow \mathsf{ct}^{(j)} + \mathsf{ct}'^{(j)} \pmod{q_j}$ for $0 \leq j \leq \ell$.

$\underline{\texttt{Mult}_{\mathsf{evk}}(\mathsf{ct}, \mathsf{ct}')}$. Given two ciphertexts $\mathsf{ct} = \left(\mathsf{ct}^{(j)} = (c_0^{(j)}, c_1^{(j)})\right)_{0 \le j \le \ell}$ and $\mathsf{ct}' = \left(\mathsf{ct}'^{(j)} = (c_0'^{(j)}, c_1'^{(j)})\right)_{0 \le j \le \ell}$, perform the following procedures and return a ciphertext $\mathsf{ct}_{\mathsf{mult}} \in \prod_{j=0}^{\ell} R_{q_j}^2$.

1. For $0 \le j \le \ell$, compute
$$\begin{aligned} d_0^{(j)} &\leftarrow c_0^{(j)} c_0'^{(j)} \pmod{q_j}, \\ d_1^{(j)} &\leftarrow c_0^{(j)} c_1'^{(j)} + c_1^{(j)} c_0'^{(j)} \pmod{q_j}, \\ d_2^{(j)} &\leftarrow c_1^{(j)} c_1'^{(j)} \pmod{q_j}. \end{aligned}$$
2. Compute $\texttt{ModUp}_{\mathcal{C}_\ell \to \mathcal{D}_\ell}(d_2^{(0)}, \ldots, d_2^{(\ell)}) = (\tilde{d}_2^{(0)}, \ldots, \tilde{d}_2^{(k-1)}, d_2^{(0)}, \ldots, d_2^{(\ell)})$.
3. Compute
$$\tilde{\mathsf{ct}} = (\tilde{\mathsf{ct}}^{(0)} = (\tilde{c}_0^{(0)}, \tilde{c}_1^{(0)}), \ldots, \tilde{\mathsf{ct}}^{(k+\ell)} = (\tilde{c}_0^{(k+\ell)}, \tilde{c}_1^{(k+\ell)})) \in \prod_{i=0}^{k-1} R_{p_i}^2 \times \prod_{j=0}^{\ell} R_{q_j}^2$$
where $\tilde{\mathsf{ct}}^{(i)} = \tilde{d}_2^{(i)} \cdot \mathsf{evk}^{(i)} \pmod{p_i}$ and $\tilde{\mathsf{ct}}^{(k+j)} = d_2^{(j)} \cdot \mathsf{evk}^{(k+j)} \pmod{q_j}$ for $0 \le i < k$, $0 \le j \le \ell$.
4. Compute
$$\begin{aligned} \left(\hat{c}_0^{(0)}, \ldots, \hat{c}_0^{(\ell)}\right) &\leftarrow \texttt{ModDown}_{\mathcal{D}_\ell \to \mathcal{C}_\ell}\left(\tilde{c}_0^{(0)}, \ldots, \tilde{c}_0^{(k+\ell)}\right), \\ \left(\hat{c}_1^{(0)}, \ldots, \hat{c}_1^{(\ell)}\right) &\leftarrow \texttt{ModDown}_{\mathcal{D}_\ell \to \mathcal{C}_\ell}\left(\tilde{c}_1^{(0)}, \ldots, \tilde{c}_1^{(k+\ell)}\right). \end{aligned}$$
5. Output the ciphertext $\mathsf{ct}_{\mathsf{mult}} = (\mathsf{ct}_{\mathsf{mult}}^{(j)})_{0 \le j \le \ell}$ where $\mathsf{ct}_{\mathsf{mult}}^{(j)} \leftarrow (\hat{c}_0^{(j)} + d_0^{(j)}, \hat{c}_1^{(j)} + d_1^{(j)}) \pmod{q_j}$ for $0 \le j \le \ell$.

We first generate an extended ciphertext $(d_0, d_1, d_2)$ that decrypts to the product of the input plaintexts under the extended secret key $(1, s, s^2)$. As mentioned before, we use the evaluation key to transform $d_2$ into a normal ciphertext. Our homomorphic multiplication algorithm is somewhat more complicated compared to the ordinary `HEAAN` because we switch the ciphertext moduli approximately using our approximate algorithms.

$\underline{\texttt{RS}(\mathsf{ct})}$. For a level-$\ell$ ciphertext $\mathsf{ct} = \left(\mathsf{ct}^{(j)} = (c_0^{(j)}, c_1^{(j)})\right)_{0 \le j \le \ell} \in \prod_{j=0}^{\ell} R_{q_j}^2$, compute $c_i'^{(j)} \leftarrow q_\ell^{-1} \cdot \left(c_i^{(j)} - c_i^{(\ell)}\right) \pmod{q_j}$ for $i = 0, 1$ and $0 \le j < \ell$. Output the ciphertext $\mathsf{ct}' \leftarrow \left(\mathsf{ct}'^{(j)} = (c_0'^{(j)}, c_1'^{(j)})\right)_{0 \le j \le \ell-1} \in \prod_{j=0}^{\ell-1} R_{q_j}^2$.

For a ciphertext $\mathsf{ct}$ encrypting a plaintext $m$, the rescaling algorithm returns an encryption of $q_\ell^{-1} \cdot m \approx q^{-1} \cdot m$ at level $(\ell - 1)$. The output ciphertext contains an additional error from the approximation of $q$ to $q_\ell$ and the rounding of the input ciphertext. The correctness of our scheme will be shown in Appendix A with noise analysis.

## 5 Software Implementation

In this section, we provide experimental results with parameter sets. In our implementation, every number is stored as an unsigned 64-bit integer, which is standard on computer system. All homomorphic operations provided in our scheme are expressed as word size operations defined on this standard variable type, so our HE library does not depend on any multi-precision numerical library. Our implementation was performed on a machine with an Intel Core i5 running at 2.9 GHz processor on a single-threaded mode, and its source code is publicly available at `https://github.com/HanKyoohyung/FullRNS-HEAAN`.

We adapt the discrete Fourier transformation to transform a polynomial represented by its coefficient vector into the vector of evaluations at primitive roots of unity modulo a prime. The modulus switching algorithms require the coefficient representation, but we can manipulate the NTT representation for arithmetic operations. Consequently, the complexity of homomorphic operations mainly depends on this transformation between two representations. We implemented the NTT conversion and its inverse based on the butterfly techniques of Cooley-Tukey [12] and Gentleman-Sande [19], respectively. We also optimized these algorithms using Montgomery modular multiplication and butterfly algorithms [26] and Barrett reduction algorithm [4].

### 5.1 Parameter Sets and Benchmark

We propose parameter sets for multiplicative depths $L$ from 5 to 15 in Table 1. It also shows experimental results for encryption, decryption, addition, scalar-multiplication, and multiplication (together with the rescaling operation) of the original implementation `HEAAN` and our RNS variant denoted by `HEAAN-RNS`.

The smallest ciphertext modulus $q_0$ should be larger than an encrypted plaintext for the correctness of the decryption circuit. We use $\log q_0 \approx 61$ and $\log q_i \approx 55$ for $i = 1, \ldots, L$. We present a list of primes in Appendix B. For a fair comparison, we choose a power-of-two integer $Q_L$ of the same bit size as the implementation of the original `HEAAN`. The coefficients of error polynomials are sampled from the discrete Gaussian distribution of standard deviation $\sigma = 3.2$ and a secret key is chosen randomly from the set of signed binary polynomials with the Hamming weight $h = 64$. We used the estimator of Albrecht at el. [2] to guarantee that the proposed parameter sets achieve at least 80-bit security level against the known attacks against the LWE problem.

Our implementation of the RNS variant improved the performance of basic operations by approximately ten times compared to the original `HEAAN` [11, 10]. For example, the encryption, decryption, addition, and multiplication are speedups of 9.1, 17.3, 7.4, and 8.3 times, respectively, when evaluating a circuit of depth $L = 10$.

In Appendix A, we analyze the growth of errors and provide theoretical upper bounds on the growth during homomorphic operations. Fig. 1 depicts the bit precisions of an encrypted plaintext during an evaluation of homomorphic

| Variant | $L$ | $N$ | $\log q$ | $\lceil \log Q_L \rceil$ | Enc (ms) | Dec (ms) | Add (ms) | Cmult (ms) | Mult&RS (ms) |
|---|---|---|---|---|---|---|---|---|---|
| HEAAN | 5 | $2^{15}$ | 55 | 336 | 332 | 106 | 30 | 204 | 740 |
| | 10 | $2^{15}$ | | 611 | 530 | 135 | 32 | 281 | 1,355 |
| | 15 | $2^{16}$ | | 886 | 1,465 | 344 | 70 | 762 | 4,169 |
| HEAAN-RNS | 5 | $2^{15}$ | 55 | 336 | 31 | 4.6 | 2.9 | 25 | 85 |
| | 10 | $2^{15}$ | | 611 | 58 | 7.8 | 4.3 | 44 | 164 |
| | 15 | $2^{16}$ | | 886 | 177 | 10.0 | 15.5 | 125 | 563 |

Table 1: Comparison of experimental results of HEAAN and HEAAN-RNS

multiplications for $L = 10$ with the parameter set in Table 1. We also provide an empirical result on the precision loss.

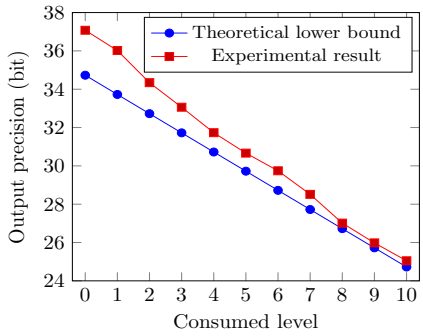


Fig. 1: Bit precision of encrypted plaintext

Our scheme exploits the approximate rounding operation which introduces an additional error. We observed that the precision of an output value is reduced by about three bits compared to the original HEAAN scheme. However, this small gap is not an critical issue in most of applications where an approximate result is sufficient for their purposes. In addition, we can easily increase the precision by setting a larger basis while still keeping advantages in the efficiency.

### 5.2 Homomorphic Evaluation of Statistical and Analytic Functions

The HEAAN scheme can evaluate an arbitrary analytic function by exploiting its polynomial approximation. Table 2 shows a parameter set and evaluation timings for the multiplicative inverse, the exponential function, and the sigmoid function $\sigma(x) = (1+\exp(-x))^{-1}$. We adapt the approximation method for multiplicative

| Function | Degree | $N$ | $\log q$ | $\lceil \log Q_L \rceil$ | Total time | Amortized time |
|---|---|---|---|---|---|---|
| $x^{-1}$ | 15 | $2^{14}$ | 55 | 281 | 167ms | 21$\mu$s |
| $\exp x$ | 7 | $2^{14}$ | 55 | 281 | 164ms | 20$\mu$s |
| Sigmoid | 7 | $2^{14}$ | 55 | 281 | 161ms | 19$\mu$s |

Table 2: Homomorphic evaluation of analytic functions

inverse of [11, Algorithm 2] and evaluate the approximate polynomial of degree 15. For the exponential and sigmoid functions, we use the Taylor expansions up to degree 7. The output ciphertexts have at least 32 bits of precision. These computations can be performed over multiple slots simultaneously, yielding a better amortized performance per slot.

We also evaluated mean and variance functions that are the most common quantities in statistical analysis. There have been a few attempts to evaluate these measurements on an HE system. For example, Lauter et al. [30] took about six seconds to obtain the square sum of 100 integers without division by the number of elements.

For computation of mean and variance of $n$ numbers, we encrypt all the numbers in a single ciphertext and compute their summation by applying the partial sum algorithm [9, Algorithm 2]. It repeats to rotate an encrypted plaintext vector and add it to the original ciphertext. The resulting ciphertext encrypts the mean value in every plaintext slot. The following example describes the partial sum algorithm when $n = 4$.

$$\begin{aligned}(m_1, m_2, m_3, m_4) &\mapsto (m_1, m_2, m_3, m_4) + (m_3, m_4, m_1, m_2) \\ &= (m_1 + m_3, m_2 + m_4, m_1 + m_3, m_2 + m_4) \\ &\mapsto \left(\sum_{i=1}^{4} m_i, \sum_{i=1}^{4} m_i, \sum_{i=1}^{4} m_i, \sum_{i=1}^{4} m_i\right)\end{aligned}$$

Contrary to previous work, the approximate HE scheme can perform a division by $n$ by multiplying the constant $\lfloor q/n \rceil$ and rescaling by one level. In the case of the variance function, we first square an input ciphertext and apply the same procedure to get a ciphertext encrypting the mean square in its plaintext slots. Then the variance of input data can be computed by subtracting the square of the encrypted mean value. We summarize the parameter and experimental results for homomorphic evaluation of statistical functions on $n = 2^{13}$ numbers in Table 3.

### 5.3 Homomorphic Training of Logistic Regression Model

The security and privacy issues have arisen on machine learning because the training of a model requires a large database consisting of sensitive information while the prediction phase is based on private information of individuals.

| | Number of elements ($n$) | $N$ | $\log q$ | $\lceil \log Q_L \rceil$ | Total time |
|---|---|---|---|---|---|
| Mean | $2^{13}$ | $2^{14}$ | 55 | 171 | 307ms |
| Variance | | | | | 518ms |

Table 3: Homomorphic evaluation of statistic functions

The technology of an HE system is a promising solution to address these issues by aggregating encrypted personal data and building a model without information leakage. ML Confidential [23] and CryptoNets [22] are notable examples of leveraging the technology of HE for secure outsourcing of machine learning applications.

In particular, `HEAAN` [11, 9] is a strong candidate for machine learning tasks since most of training and prediction algorithms contain an arithmetic over the real numbers. For example, iDASH Security and Privacy Competition in 2017[4] announced a task which aims to build a logistic regression model from homomorphically encrypted genomic data. To be precise, for a given dataset consisting of $n$ samples $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{\pm 1\}$ of $d$ features and a binary class, the goal was to find a weight vector $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$ which minimizes the loss function

$$J(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log(1 + \exp(-\boldsymbol{\beta}^T \boldsymbol{z}_i))$$

where $\boldsymbol{z}_i = y_i \cdot (1, \boldsymbol{x}_i)$ for $1 \le i \le n$. The best solution [27] adapted the `HEAAN` library [10] to evaluate Nesterov's accelerated gradient descent method [31].

We implemented the same algorithm based on `HEAAN-RNS` to show its versatility and efficiency. For a fair comparison, we adapt the previous encoding and evaluation strategies: the whole database is encrypted in a single ciphertext and the sigmoid function of the gradient descent algorithm is approximated to its least squares approximation. Our implementation took about 1.8 minutes to train a model based on Low Birth Weight Study (lbw) [29] and Umaru Impact Study (uis) [33] datasets using a single core processor, compared to 3.5 minutes of the previous best solution [27] using four cores, while maintaining the accuracy and area under the ROC curve (AUC) of the resulting classifier.

| Dataset | Num of features | Num of samples | Num of iterations | $N$ | $\log q$ | $\lceil \log Q_L \rceil$ | Total time | Accuracy | AUC |
|---|---|---|---|---|---|---|---|---|---|
| lbw | 9 | 189 | 5 | $2^{16}$ | 40 | 1061 | 1.82min | 69.73% | 0.62 |
| uis | 8 | 575 | 5 | $2^{16}$ | 40 | 1061 | 1.83min | 74.43% | 0.59 |

Table 4: Homomorphic training of logistic regression model

[4] `http://www.humangenomeprivacy.org/2017/`

## 6 Conclusions and Future Work

In this article, we demonstrate a variant of `HEAAN` based on the RNS representation of polynomials. In the previous implementation, ciphertext moduli were selected as powers of a fixed base for the correctness of rescaling process. We resolve the issue by taking an RNS basis consisting of primes close to the base integer. In addition, we propose variants of modulus switching algorithms which can be computed without any RNS conversion or multi-precision arithmetic.

One disadvantage of our method is that it makes a trade-off between performance and accuracy. Because of the approximation error of an RNS basis, our scheme may have less accuracy compared to the original scheme when using the same parameter. Recently, `SEAL` version 3.0 [32] has been released. It supports a full RNS variant of `HEAAN`, which is slightly different from our scheme. The main difference is that a ciphertext of `SEAL` contains a scaling factor which can be changed during computation. In other words, it continuously tracks the computation and updates the scaling factor information. This method does not have the above accuracy issue, but it is less intuitive and causes new problems related to the management of scaling factors. For example, the addition (resp. multiplication) of ciphertexts of different scaling factors (resp. levels) requires pre (resp. post) processing. It would be an interesting future work to combine the two methods to design a new scheme with enhanced functionality and flexibility.

## Acknowledgments

This work was partially supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government(MSIT)(No.2017R1A5A1015626). M. Kim was supported by the National Institute of Health (NIH) under award number U01EB023685 and R01GM118574 as well as Cancer Prevention Research Institute of Texas (CPRIT) grant RR180012.

## References

1. C. Aguilar-Melchor, J. Barrier, S. Guelton, A. Guinet, M.-O. Killijian, and T. Lepoint. NFLlib: NTT-based fast lattice library. In *Cryptographers Track at the RSA Conference*, pages 341–356. Springer, 2016.
2. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.
3. J.-C. Bajard, J. Eynard, M. A. Hasan, and V. Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. In *International Conference on Selected Areas in Cryptography*, pages 423–442. Springer, 2016.
4. P. Barret. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. *Advances in Cryptology - CRYPTO 1986*, 1986.
5. C. Bonte, C. Bootland, J. W. Bos, W. Castryck, I. Iliashenko, and F. Vercauteren. Faster homomorphic function evaluation using non-integral base encoding. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 579–600. Springer, 2017.

6. Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology–CRYPTO 2012*, pages 868–886. Springer, 2012.
7. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proc. of ITCS*, pages 309–325. ACM, 2012.
8. H. Chen, K. Laine, R. Player, and Y. Xia. High-precision arithmetic in homomorphic encryption. In *Cryptographers Track at the RSA Conference*, pages 116–136. Springer, 2018.
9. J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. Bootstrapping for approximate homomorphic encryption. In *Advanced in Cryptology–EUROCRYPT 2018*, pages 360–384. Springer, 2018.
10. J. H. Cheon, A. Kim, M. Kim, and Y. Song. Implementation of HEAAN, 2016. `https://github.com/kimandrik/HEAAN`.
11. J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017*, pages 409–437. Springer, 2017.
12. J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.
13. A. Costache and N. P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In *Cryptographers Track at the RSA Conference*, pages 325–340. Springer, 2016.
14. A. Costache, N. P. Smart, and S. Vivek. Faster homomorphic evaluation of discrete fourier transforms. In *International Conference on Financial Cryptography and Data Security*, pages 517–529. Springer, 2017.
15. A. Costache, N. P. Smart, S. Vivek, and A. Waller. Fixed-point arithmetic in she schemes. In *International Conference on Selected Areas in Cryptography*, pages 401–422. Springer, 2016.
16. M. v. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2010*, pages 24–43. Springer, 2010.
17. L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology–EUROCRYPT 2015*, pages 617–640. Springer, 2015.
18. J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `https://eprint.iacr.org/2012/144`.
19. W. M. Gentleman and G. Sande. Fast fourier transforms: for fun and profit. In *Proceedings of the November 7-10, 1966, fall joint computer conference*, pages 563–578. ACM, 1966.
20. C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178. ACM, 2009.
21. C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology–CRYPTO 2012*, pages 850–867. Springer, 2012.
22. R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.
23. T. Graepel, K. Lauter, and M. Naehrig. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pages 1–21. Springer, 2012.

24. S. Halevi, Y. Polyakov, and V. Shoup. An improved RNS variant of the BFV homomorphic encryption scheme. Cryptology ePrint Archive, Report 2018/117, 2018. https://eprint.iacr.org/2018/117.
25. S. Halevi and V. Shoup. Design and implementation of a homomorphic-encryption library. *IBM Research (Manuscript)*, 2013.
26. D. Harvey. Faster arithmetic for number-theoretic transforms. *Journal of Symbolic Computation*, 60:113–119, 2012.
27. A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon. Logistic regression model training based on the approximate homomorphic encryption. *BMC Medical Genomics*, 11(4):83, 2018.
28. M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang. Secure logistic regression based on homomorphic encryption: design and evaluation. *JMIR medical informatics*, 6(2), 2018.
29. lbw. Low birth weight study data. https://rdrr.io/rforge/LogisticDx/man/lbw.html, 2017.
30. M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.
31. Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.
32. Simple Encrypted Arithmetic Library (release 3.0.0). http://sealcrypto.org, Oct. 2018. Microsoft Research, Redmond, WA.
33. uis. Umaru impact study data. https://rdrr.io/rforge/LogisticDx/man/uis.html, 2017.

# A Correctness and Noise Estimation

Our improved HE scheme is based on two main techniques- approximate basis and modulus switching, and both of them induce some additional errors. In this section, we estimate the size of errors and show that they can be managed by choosing a proper HE parameter set. For convenience, we adapt the same notations as in Section 4.

## A.1 Approximate Modulus Switching

**Fast Conversion.** Our modulus switching algorithms are based on the fast basis conversion algorithm introduced in [3]. For the RNS representation $[a]_{\mathcal{C}}$ of an integer $a \in \mathbb{Z}_{Q_\ell}$, the fast conversion algorithm $\texttt{Conv}_{\mathcal{C}\to\mathcal{B}}([a]_{\mathcal{C}})$ computes the RNS representation of $a' = \sum_{j=0}^{\ell-1}[a^{(j)} \cdot \hat{q}_j^{-1}]_{q_j} \cdot \hat{q}_j$ with respect to the basis $\mathcal{B}$. Then there exists an integer $e \in [-\ell/2, \ell/2]$ satisfying $a' = a + Q \cdot e$ since $a' \equiv a \pmod{Q}$ and $|a'| \leq (\ell/2) \cdot Q$.

**Approximate Modulus Raising.** Let $[a]_{\mathcal{C}}$ be the RNS representation of an integer $a \in \mathbb{Z}_Q$. The approximate modulus raising algorithm $\texttt{ModUp}_{\mathcal{C}\to\mathcal{D}}([a]_{\mathcal{C}})$ returns the concatenation of $\texttt{Conv}_{\mathcal{C}\to\mathcal{B}}([a]_{\mathcal{C}})$ and $[a]_{\mathcal{C}}$, which is the RNS representation of $a + Q \cdot e$ for some integer $e \in [-\ell/2, \ell/2]$ from the property of the fast conversion algorithm.

**Approximate Modulus Reduction.** Let $[\tilde{b}]_{\mathcal{D}} = (\tilde{b}^{(i)})$ for $0 \le i \le k+\ell-1$ be the RNS representation of an integer $\tilde{b} \in \mathbb{Z}_{P \cdot Q}$. It satisfies that $(\tilde{b}^{(0)}, \ldots, \tilde{b}^{(k-1)}) = [a]_{\mathcal{B}}$ for $a = [\tilde{b}]_P$. From the property of the fast conversion algorithm, we have that $(\tilde{a}^{(0)}, \ldots, \tilde{a}^{(\ell-1)}) \leftarrow \mathtt{Conv}_{\mathcal{B} \to \mathcal{C}}([a]_{\mathcal{B}})$ is the RNS representation of $\tilde{a} := a + P \cdot e$ for some integer $e$ such that $|\tilde{a}| \le (k/2) \cdot P$.

Let $b = P^{-1} \cdot (\tilde{b} - \tilde{a})$. It is an integer from $\tilde{b} \equiv a \equiv \tilde{a} \pmod{P}$. Then the output of $\mathtt{ModDown}_{\mathcal{D} \to \mathcal{C}}([\tilde{b}]_{\mathcal{D}})$ is equal to $[b]_{\mathcal{C}}$ since $b \equiv P^{-1} \cdot (\tilde{b} - \tilde{a}) \equiv \left(\prod_{i=0}^{k-1} p_i\right)^{-1} \cdot (\tilde{b}^{(k+j)} - \tilde{a}^{(j)}) \pmod{q_j}$. Note that the integer $b \in \mathbb{Z}_Q$ satisfies $|b - P^{-1} \cdot \tilde{b}| = P^{-1} \cdot |\tilde{a}| \le k/2$.

### A.2 Homomorphic Operations

In this paragraph, we will focus on homomorphic operations provided in our scheme. We define $\|a\|_\infty$ and $\|a\|_1$ by the relevant norms on the coefficients vector $(a_0, \ldots, a_{N-1})$ of $a(X)$. Let $\zeta = \exp(-\pi i/N)$. Recall that the canonical embedding map on $K = \mathbb{Q}[X]/(X^n+1)$ is defined by $a(X) \mapsto (a(\zeta), a(\zeta^3), \ldots, a(\zeta^{2N-1}))$. Its $\ell_\infty$-norm is called the canonical embedding norm, and denoted by $\|a\|_\infty^{\mathsf{can}} = \|\sigma(a)\|_\infty$. Note that $\|a\|_\infty^{\mathsf{can}} = \|\tau(a)\|_\infty$ for the decoding map $\tau$ and for any $a \in K$.

We specify the distributions $\chi_{\mathsf{key}}$, $\chi_{\mathsf{err}}$, and $\chi_{\mathsf{enc}}$ for noise analysis of our scheme. For an positive integer $h$, the secret key distribution $\chi_{\mathsf{key}}$ follows a uniform distribution over the set of signed binary vectors in $\{0, \pm 1\}^N$ whose Hamming weight (the number of nonzero coefficients) is exactly $h$. The error distribution $\chi_{\mathsf{err}}$ chooses a polynomial $s$ by sampling its coefficients independently from the discrete Gaussian distribution of variance $\sigma^2$ for a real $\sigma > 0$. The encryption key distribution $\chi_{\mathsf{enc}}$ draws each entry in the vector from $\{0, \pm 1\}$, with probability $1/4$ for each of $+1$ and $-1$, and probability being zero $1/2$.

We follow the same methodology for noise estimation as in [21, 13, 11]. Assume that a polynomial $a(X)$ is sampled from one of the distributions used in our HE scheme. Since $a(\zeta)$ is the inner product of coefficient vector of $a$ and the fixed vector $(1, \zeta, \ldots, \zeta^{N-1})$ of Euclidean norm $\sqrt{N}$, the random variable $a(\zeta)$ has variance $V_{\mathsf{err}} = \sigma^2 \cdot N$, where $\sigma^2$ is the variance of each coefficient of $a$. Similarly, $a(\zeta)$ a the variance of $V_q = q^2 N/12$ (resp. $N/2$), when $a$ is sampled from $U(R_q)$ (resp. $\chi_{\mathsf{enc}}$). In particular, it has variance $h$ when $a(X)$ is chosen from $\chi_{\mathsf{key}}$. Moreover, we can assume that $a(\zeta)$ is distributed similarly to a Gaussian random variable over complex plane since it is a sum of many independent and identically distributed random variables. Every evaluations at root of unity $\zeta^j$ share the same variance. Hence, we will use $6 \cdot \sqrt{V}$ as a high-probability bound on the canonical embedding norm of $a$ when each coefficient has a variance $V$. For a multiplication of two independent random variables close to Gaussian distributions with variances $V_1$ and $V_2$, we will use a high-probability bound $16 \cdot \sqrt{V_1 V_2}$.

**Encryption.** Our encryption algorithm does not use any approximate modulus switching algorithms. Therefore, it has exactly the same noise with the original implementation of `HEAAN` scheme. For a plaintext $m \in R$, it returns a ciphertext

$\mathsf{ct} \in R_{Q_L}^2$ which satisfies $\langle \mathsf{ct}, \mathsf{sk} \rangle \equiv m + e \pmod{Q_L}$ for some $e \in R$ such that $\|e\|_\infty^{\mathsf{can}} \le B_{\mathsf{enc}} = 8\sqrt{2}\sigma N + 6\sigma\sqrt{N} + 16\sigma\sqrt{hN}$ from Lemma 1 of [11].

**Addition.** It does not induce any additional error since $\langle \mathsf{ct}_{\mathsf{add}}, \mathsf{sk} \rangle \equiv \langle \mathsf{ct}, \mathsf{sk} \rangle + \langle \mathsf{ct}', \mathsf{sk} \rangle \pmod{Q_\ell}$.

**Rescaling.** Let $\mathsf{ct} = \left(\mathsf{ct}^{(j)} = (c_0^{(j)}, c_1^{(j)})\right)_{0 \le j \le \ell} \in \prod_{j=0}^{\ell} R_{q_j}^2$ be an input ciphertext of level $\ell$, and $\mathsf{ct}' \leftarrow \left(\mathsf{ct}'^{(j)} = (c_0'^{(j)}, c_1'^{(j)})\right)_{0 \le j \le \ell-1} \leftarrow \mathtt{RS}(\mathsf{ct})$ be the output ciphertext obtained by $c_i'^{(j)} \leftarrow q_\ell^{-1} \cdot (c_i^{(j)} - c_i^{(\ell)})$ for $i = 0, 1$ and $0 \le j < \ell$.

Let $c_i \in R_{Q_L}$ be the polynomials satisfying $[c_i]_{\mathcal{C}_\ell} = (c_i^{(0)}, \ldots, c_i^{(\ell)})$ for $i = 0, 1$. Then we have that $[c_i]_{\mathcal{C}_{\ell-1}} = (c_i'^{(0)}, \ldots, c_i'^{(\ell-1)})$ for $c_i' = q_\ell^{-1} \cdot (c_i - [c_i]_{q_\ell}) = \lfloor q_\ell^{-1} \cdot c_i \rceil$, that is, our rescaling procedure computes the exactly same ciphertext as in the original `HEAAN` scheme with RNS representation. Therefore, we have $[\langle \mathsf{ct}', \mathsf{sk} \rangle]_{Q_{\ell-1}} = q_\ell^{-1} \cdot [\langle \mathsf{ct}, \mathsf{sk} \rangle]_{Q_\ell} + e_{\mathsf{rs}}$ for some $e_{\mathsf{rs}} \in K$ satisfying $\|e\|_\infty^{\mathsf{can}} \le B_{\mathsf{rs}} = \sqrt{N/3} \cdot (3 + 8\sqrt{h})$ from Lemma 2 of [11].

**Multiplication.** Suppose that we are given two level-$\ell$ ciphertext $\mathsf{ct}$ and $\mathsf{ct}'$. The output of the first step in the multiplication algorithm is the RNS representation of $(d_0, d_1, d_2) \in R_{Q_\ell}^3$ such that $d_0 + d_1 \cdot s + d_2 \cdot s^2 \equiv \langle \mathsf{ct}, \mathsf{sk} \rangle \cdot \langle \mathsf{ct}', \mathsf{sk} \rangle \pmod{Q_\ell}$. The output of the second step is the RNS representation of $\tilde{d}_2 = d_2 + Q_\ell \cdot e$ with respect to the basis $\mathcal{D}_\ell$ for some $e \in R$ satisfying $\|\tilde{d}_2\|_\infty \le \frac{1}{2}(\ell+1) \cdot Q_\ell$. We may assume that the integral polynomial $\tilde{d}_2$ behaves like the sum of $(\ell+1)$ independent and uniform random variables over $R_{Q_\ell}$, so its variance is $V = \frac{1}{2}(\ell+1) \cdot (Q_\ell^2 \cdot N/12)$.

Since the first $(k+\ell+1)$ components of the evaluation key $\mathsf{evk}$ can be viewed as an encryption of $P \cdot s^2$ modulo $P \cdot Q_\ell$, the output $\tilde{\mathsf{ct}}$ of the third step is an encryption of $P \cdot \tilde{d}_2 \cdot s^2 \equiv P \cdot d_2 \cdot s^2 \pmod{P \cdot Q_\ell}$. Its error is bounded by $16 \cdot \sqrt{V} \cdot \sqrt{N\sigma^2} = 8\sqrt{(\ell+1)/6} \cdot Q_\ell \cdot \sigma N = \sqrt{(\ell+1)/2} \cdot B_{\mathsf{ks}} \cdot Q_\ell$.

The fourth step reduces the modulus of $\tilde{\mathsf{ct}}$ using the modulus reduction algorithm. It returns a ciphertext $\hat{\mathsf{ct}} \in R_{Q_\ell}^2$ such that $P \cdot \hat{\mathsf{ct}} \approx \tilde{\mathsf{ct}}$. The error $P \cdot \hat{\mathsf{ct}} - \tilde{\mathsf{ct}}$ behaves as if it is a sum of $k$ independent and uniform random variables on $R_P$, so its variance is $k \cdot V_P = k \cdot P^2 N/12$. Finally, dividing by $P$, we obtain the error after modulus reduction. Therefore, $\hat{\mathsf{ct}}$ is an encryption of $d_2 \cdot s^2$ with an error bounded by $\sqrt{(\ell+1)/2} \cdot P^{-1} \cdot B_{\mathsf{ks}} \cdot Q_\ell + \sqrt{k} \cdot B_{\mathsf{rs}}$.

## B List of Primes

A ciphertext modulus is chosen to be a product of distinct primes and each of them satisfies the following conditions:

$$\begin{cases} |2^{-\kappa} \cdot q_j - 1| < 2^{-\eta}, \\ q_j \equiv 1 \pmod{2N}, \end{cases}$$

for some integers $\kappa$, $\eta$, and $N$. In other words, $q_j$ is an approximation of $2^\kappa$ with $\eta$-bit precision, and there is a $(2N)$-th primitive root of unity modulo $q_j$. All

primes are expressed using hexadecimal system to show how close they are to powers of two.

There are 22 primes including $q_0 = \texttt{0x20000000000b0001}$ satisfying these conditions for $\kappa = 61$, $\eta = 37$, and $N = 2^{15}$. We have 33 primes when $(\kappa, \eta, N) = (55, 31, 2^{15})$, and 26 prime numbers when $(\kappa, \eta, N) = (49, 25, 2^{15})$. The following is a list of 15 primes (among 33 primes for the second parameter) that were used in the implementation described in Table 1.

$$
\begin{aligned}
[&\texttt{80000000080001}, \texttt{80000000130001}, \texttt{7fffffffe90001},\\
&\texttt{80000000190001}, \texttt{800000001d0001}, \texttt{7fffffffbf0001},\\
&\texttt{7fffffffbd0001}, \texttt{80000000440001}, \texttt{7fffffffba0001},\\
&\texttt{80000000490001}, \texttt{80000000500001}, \texttt{7fffffffaa0001},\\
&\texttt{7fffffffa50001}, \texttt{800000005e0001}, \texttt{7fffffff9f0001}]
\end{aligned}
$$