Multi-dimensional Packing for HEAAN for Approximate Matrix Arithmetics

Jung Hee Cheon[1], Andrey Kim[1], Donggeon Yhee[1]

Seoul National University, Republic of Korea
{jhcheon, kimandrik, dark0926}@snu.ac.kr

Abstract. HEAAN is a homomorphic encryption (HE) scheme for approximate arithmetics. Its vector packing technique proved its potential in cryptographic applications requiring approximate computations, including data analysis and machine learning.
In this paper, we propose MHEAAN - a generalization of HEAAN to the case of a tensor structure of plaintext slots. Our design takes advantage of the HEAAN scheme, that the precision losses during the evaluation are limited by the depth of the circuit, and it exceeds no more than one bit compared to unencrypted approximate arithmetics, such as floating point operations. Due to the multi-dimensional structure of plaintext slots along with rotations in various dimensions, MHEAAN is a more natural choice for applications involving matrices and tensors. We provide a concrete two-dimensional construction and show the efficiency of our scheme on several matrix operations, such as matrix multiplication, matrix transposition, and inverse.
As an application, we implement the first non-interactive Deep Neural Network (DNN) classification algorithm on encrypted data and encrypted model. Due to our efficient bootstrapping, the implementation can be easily extended to DNN structure with an arbitrary number of hidden layers.

Keywords. Homomorphic encryption, approximate arithmetics, matrix operations, bootstrapping, linear transformation, encrypted deep neural network.

1 Introduction

Homomorphic encryption (HE) [39] allows to perform certain arithmetics operations in encrypted state. Following Gentry's blueprint [23], a numerous HE schemes have been proposed [17, 6, 7, 4, 5, 24, 33, 2, 25, 15, 13, 19, 18, 12, 14]. The most asymptotically efficient HE schemes are based on the hardness of RLWE, and they normally have a common structure of ciphertexts with noised encryption for security.

With the growth of more complex algorithms, such as deep learning and recommendation systems which require lots of matrix operations, the possibility of performing matrix operations is becoming crucial for homomorphic encryptions. Despite the diversity of HE schemes that achieve a variety of circuit evaluations, practical matrix operations such as matrix multiplications is still a problem in HE.

Matrix operations with HE Some works by Duong et al. [20] suggested a method for a single matrix multiplication using special packing methods, however its packing structure seems to have problems in extensions to more complex computations. A following work by Mishra et al. [35] extended [20] to several matrix multiplications but it requires an exponentially large degree of the base ring so it not seems to be practical. In another work Hiromasa et al. [30] constructed a matrix version of GSW scheme, but it considers only binary matrix cases and requires a lot of matrix multiplications. Thus it neither seems to be practical.

Recently Cheon et. al. [12] presented a method of constructing an HE scheme for arithmetics of approximate numbers (called `HEAAN`). The idea of the construction is to treat encryption noise as a part of error occurring during approximate computations. In other words, a ciphertext ct of a plaintext $m \in \mathcal{R}$ encrypted by a secret key sk for an underlying ciphertext modulus q will have a decryption structure of the form $\langle \mathsf{ct}, \mathsf{sk} \rangle = m + e \pmod{\mathcal{R}/q\mathcal{R}}$ for some small error e. `HEAAN` is based on an RLWE structure over a power-of-two $M = 2 \cdot N$ cyclotomic ring modulo q, $\mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[X]/(X^N + 1)$. A vector of complex values of size up to $N/2$ can be encoded using a variant of canonical embedding map.

`HEAAN` showed its potential by providing the winning solution of Track 3 (Homomorphic encryption based logistic regression model learning) at the iDASH privacy and security competition in 2017 [31]. In their solution authors packed a matrix of inputs in a vector. Even though the authors could provide all computations using matrix to vector packing in that particular task, due to absence of row-wise matrix rotation functionality they had to circumvent and consume an additional level during the computations.

1.1 Our contributions.

In this paper we suggest a generalization of `HEAAN` with a tensor packing method, along with natural rotations in various dimensions, which is, called the hypercube structure, also applied in HElib [26, 27, 28]. The straightforward attempt could be based on the Multivariate RLWE (m-PLWE) problem as an underlying hardness problem, introduced by Pedrouzo-Ulloa et al. [36, 37] as a multivariate variant of RLWE problem with an underlying ring $\mathbb{Z}[x_0, x_1]/(x_0^{N_0} + 1, x_1^{N_1} + 1)$ where both N_0 and N_1 are powers-of-two. However this problem succumbs to the following evaluation attack: without loss of generality assume $N_0 \geq N_1$, and substitute $x_1 = x_0^{N_0/N_1}$, then the RLWE problem over $\mathbb{Z}[x_0, x_1]/(x_0^{N_0} + 1, x_1^{N_1} + 1)$ reduces to a problem over $\mathbb{Z}[x_0]/(x_0^{N_0} + 1)$.

Secure scheme So instead, we provide a scheme `MHEAAN` based on the m-PLWE problem with indeterminates x_0 and x_1 (or in general case $x_0, \ldots, x_s$) satisfying relations given by cyclotomic polynomials corresponding to relatively prime orders. The hardness of the m-PLWE problem over this ring is shown to have reduction from the origina RLWE problem. MHEAAN enjoys all the benefits of HEAAN such as a rescaling procedure, which enables us to preserve the precision of the message after approximate computations and to reduce the size of ciphertexts significantly. Thus, the scheme can be a reasonable solution for approximate computation over the complex values. Moreover, with a multivariable structure of m-PLWE, we provide a general technique for tensor plaintext slots packing in a single ciphertext. We provide a concrete two-dimensional construction which supports matrix operations as well as standard HE operations.

Performance on matrix operations. For two-dimensional case corresponding to natural matrix structure of plaintext slots, matrix multiplication in MHEAAN is achieved in very simple way using Fox matrix multiplication algorithm [22]. In contrast to the method of Mishra et al. [35] our method does not require exponentially large degree of the base ring and we can use matrix multiplication as a part of more complex algorithms. The matrix size is also not a problem, as our method preserves matrix structure, and can combined with divide-and-conquer algorithm. Moreover MHEAAN enjoys other matrix related operations, like matrix transposition. We examined MHEAAN's performance on several matrix operations. It takes about 11 seconds for homomorphic multiplication of two 64×64 complex matrices, about 88 seconds to evaluate 16-th power of a 64×64 complex matrix, and about 2 seconds for homomorphic transposition of a 64×64 matrix. Matrix inverse is a more complicated problem even in an unencrypted case [41, 40]. We provided a method to evaluate the inverse of a matrix for a special case. It takes about 2 minutes for a 64×64 complex matrix.

Performance on bootstrapping. MHEAAN supports faster bootstrapping procedure than that of HEAAN when number of slots is sufficiently large. For base ring degree N, the bootstrapping procedure for large number of slots in MHEAAN is approximately requires $O(N^{\frac{1}{2(s+1)}})$ of ciphertext rotations and $O(N^{\frac{1}{s+1}})$ of constant multiplications where $s+1$ is the number of factors of base ring. The original HEAAN requires about $O(\sqrt{N})$ of ciphertext rotations and $O(N)$ of constant multiplications. In our implementation s is equal to 1 and the degree of ring is factored into values close to $\sqrt{N}$, so the bootstrapping complexity is reduced from $O(\sqrt{N})$ to $O(\sqrt[4]{N})$ rotations and from $O(N)$ to $O(\sqrt{N})$ constant multiplications. In practice this difference appears to be huge. For example for 2^{15} number of slots HEAAN bootstrapping takes about 24 hours to complete. On the other side for MHEAAN it takes about 2.5 mins.

Non-interactive DNN Classification. In Section 7, we apply our design to non-interactive DNN classification algorithm. In our approach not only the input data but also the model privacy is protected. The encrypted predictions achieve high accuracy 97.7% 97.8% which is similar to the accuracy 97.9% of the predictions on the plain data. Our approach is flexible and can be generalized to the DNN architecture with large number of hidden layers.

Previous implementations of encrypted prediction [3, 29] are done over the plain models, and limited number of hidden layers. The result of [3] has an impressive performance, but it is restricted only for a binary model, and is expected to have huge drowning in the efficiency when expanding to a non-binary model.

2 Preliminaries

To avoid an ambiguity, we define tensors following linear algebras :

Definition 1. *A* tensor *is an assign from a multi-indices set to values. A tensor is of rank k if the multi-indices set consists of k-tuple of indices. A vector is a rank 1 tensor and a matrix is a rank 2 tensor.*

2.1 Basic Notations

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. For vectors $\mathbf{a}$ and $\mathbf{b}$ we denote by $\langle \mathbf{a}, \mathbf{b} \rangle$ the usual dot product. We denote matrices by bold capital letters, e.g. $\mathbf{A}$, and general tensors by $\hat{\mathbf{a}}$. For a real number r, $\lfloor r \rceil$ is the nearest integer to r, rounding upwards in case of a tie. For an integer q, we identify the ring $\mathbb{Z}_q$ with $(-q/2, q/2]$ as a representative interval and for integer r we denote by $\lfloor r \rceil_q$ the reduction of r modulo q into that interval. We use $a \leftarrow \chi$ to denote the sampling a according to a distribution χ. If χ is a uniform distribution on a set $\mathcal{D}$, we use $a \leftarrow \mathcal{D}$ rather than $a \leftarrow \chi$. For rank k tensors $\hat{\mathbf{a}}, \hat{\mathbf{b}} \in \mathbb{C}^{n_1 \times \cdots \times n_k}$ we denote a component-wise product by $\hat{\mathbf{a}} \odot \hat{\mathbf{b}}$. For vectors $\mathbf{r} = (r_1, \ldots, r_k)$ and $\mathbf{g} = (g_1, \ldots, g_k)$ we denote by $\mathbf{g}^{\mathbf{r}} = (g_1^{r_1}, \ldots, g_k^{r_k})$ component powers, and by $\mathsf{rt}(\hat{\mathbf{a}}, \mathbf{r})$ a tensor obtained from $\hat{\mathbf{a}}$ by cyclic rotation by r_i in corresponding index i. For example, in case of matrices i.e. rank 2 tensors, we have:

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n_1-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_0-1,0} & a_{n_0-1,1} & \cdots & a_{n_0-1,n_1-1} \end{bmatrix}$$

$$\mathsf{rt}(\mathbf{A}, (r_0, r_1)) = \begin{bmatrix} a_{r_0,r_1} & a_{r_0,r_1+1} & \cdots & a_{r_0,r_1-1} \\ a_{r_0+1,r_1} & a_{r_0+1,r_1+1} & \cdots & a_{r_0+1,r_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r_0-1,r_1} & a_{r_0-1,r_1+1} & \cdots & a_{r_0-1,r_1-1} \end{bmatrix}$$

where indices are taken modulus n_i. Denote the security parameter throughout the paper: all known valid attacks against the cryptographic scheme under scope should take bit operations.

2.2 The Cyclotomic Ring and Canonical Embedding

For an integer M consider its decomposition into primes $M = 2^{k_0} \cdot p_1^{k_1} \cdot \dots \cdot p_s^{k_s} = \prod_{i=0}^{s} M_i$, where $M_0 = 2^{k_0}$, and $M_i = p_i^{k_i}$ for $i = 1, \dots, s$. We will consider the cases $k_0 > 2$. Let $N_i = \phi(M_i) = (1 - \frac{1}{p_i})M_i$ for $i = 0, \dots, s$, and $N = \phi(M) = \prod_{i=0}^{s} N_i$. Denote tensors $\hat{\mathbf{N}} = N_0 \times N_1 \times \dots \times N_s$, $\hat{\mathbf{N}}_h = N_0/2 \times N_1 \times \dots \times N_s$, and vectors $\mathbf{N} = (N_0, N_1, \dots, N_s)$, $\mathbf{N}_h = (N_0/2, N_1, \dots, N_s)$. Let $\Phi_M(x)$ be M-th cyclotomic polynomial. Let $\mathcal{R} = \mathbb{Z}[x]/\Phi_M(x)$ and $\mathcal{S} = \mathbb{R}[x]/\Phi_M(x)$. The canonical embedding τ_M of $a(x) \in \mathbb{Q}[x]/(\Phi_M(x))$ into $\mathbb{C}^N$ is the vector of evaluation values of $a(x)$ at the roots of $\Phi_M(x)$. We naturally extend it to the set of real polynomials $\mathcal{S}$, $\tau_M : \mathcal{S} \to \mathbb{C}^N$, so $\tau_M(a(x))$ will be defined as $(a(\xi_M^j))_{j \in \mathbb{Z}_M^\star} \in \mathbb{C}^N$ for any $a \in \mathcal{R}$ where $\xi_M = \exp(-2\pi i/M)$ is a primitive M-th root of unity. The ℓ_∞-norm of $\tau_M(a(X))$ is called the *canonical embedding norm* of a, denoted by $\|a\|_\infty^{\mathsf{can}} = \|\tau_M(a)\|_\infty$. The canonical embedding norm $\|\cdot\|_\infty^{\mathsf{can}}$ satisfies the following properties:

- For all $a, b \in \mathcal{R}$, we have $\|a \cdot b\|_\infty^{\mathsf{can}} \le \|a\|_\infty^{\mathsf{can}} \cdot \|b\|_\infty^{\mathsf{can}}$
- For all $a \in \mathcal{R}$, we have $\|a\|_\infty^{\mathsf{can}} \le \|a\|_1$.
- For all $a \in \mathcal{R}$, we have $\|a\|_\infty \le \|a\|_\infty^{\mathsf{can}}$.

Refer [16] for more details.

2.3 m-PLWE Problem

Here we set up an underlying hardness problem.

Proposition 1. *If $M_0, M_1, \cdots, M_s$ are pairwisely coprime, then there is a ring isomorphism*

$$\mathcal{S} = \mathbb{R}[x]/\Phi_M(x) \cong \mathbb{R}[x_0, \dots, x_s]/(\Phi_{M_0}(x_0), \dots \Phi_{M_s}(x_s)) = \mathcal{S}'$$

and the map induces a ring isomorphism

$$\mathcal{R} = \mathbb{Z}[x]/\Phi_M(x) \cong \mathbb{Z}[x_0, \dots, x_s]/(\Phi_{M_0}(x_0), \dots \Phi_{M_s}(x_s)) = \mathcal{R}'.$$

We refers [34] for RLWE-problem.

Definition 2. *A decisional RLWE problem $\mathsf{RLWE}_{\mathcal{R},\sigma}$ is a distinguishing problem between uniform distribution $(a(x), b(x))$ and a distribution $(a(x), a(x)s(x) + e(x))$ such that $a(x), b(x), s(x) \leftarrow \mathcal{R}/q\mathcal{R}$ and $e(x)$ is given by the image of a sample in $\mathcal{R}$ whose canonical embedding has components following a Gaussian distribution of variance σ^2 independently.*

Definition 3. *A decisional multivariate PLWE problem $\mathsf{m\text{-}PLWE}_{\mathcal{R}',\sigma'}$ is a distinguishing problem between uniform distribution $(a(\mathbf{x}), b(\mathbf{x}))$ and a distribution $(a(\mathbf{x}), a(\mathbf{x})s(\mathbf{x}) + e(\mathbf{x}))$ such that $a(\mathbf{x}), b(\mathbf{x}), s(\mathbf{x}) \in \mathcal{R}'/q\mathcal{R}'$ and $e(\mathbf{x})$ is given by the image of a sample in $\mathcal{R}'$ whose coefficients follow a Gaussian distribution of variance σ'^2 independently.*

Since the rings are not power-of-2 cyclotomic, their images under Minkowski embedding looks non-spherical at first glance. If they are, the distinguishing problems could be vulnerable attacks such as [21, 8, 38]. Fortunately, the attacks are valid only when the base ring is not Galois. In particular, in the attack model the embedded images are not spherical because the defining equation of the rings do not completely split modulus q.

The m-PLWE problem is suspected to be weak under evaluation attacks such as in case of the ring $\mathbb{Z}[x_0, x_1]/(\Phi_{M_0}(x_0), \Phi_{M_1}(x_1))$ for the powers-of-two M_0, M_1. The attack also seems to be expanding at least partially to the case $\gcd(M_i, M_j) > 1$. We design our scheme using relatively prime M_i's to avoid this case. Further we show the hardness of our case by devising a reduction from the original RLWE problem to m-PLWE problem with relatively prime M_i's.

Lemma 1. *(Hardness of m-PLWE) Let $\mathcal{R}$ and $\mathcal{R}'$ be given as proposition 1. Then $\mathsf{RLWE}_{\mathcal{R},\sigma}$ reduces to $\mathsf{m\text{-}PLWE}_{\mathcal{R}',c\sigma}$, where $c^2 = \prod_{i=1}^{s}\left(\frac{p_i-1}{p_i} \times (2 + \frac{12\pi}{p_i})\right)$.*

In particular, c is less than $\sqrt{3}$ if $p_i \geq 41 > 12\pi$ or $p_i = 3, 37$. As p increases, c tends to be $\sqrt{2}$. The followings are approximations of c :

$$(p_i, c) = (5, 2.8), (7, 2.6), (11, 2.3), (13, 2.2), (17, 2.0),$$
$$(19, 2.0), (23, 1.9), (29, 1.8), (31, 1.9)$$

For $p_i = 3$ and 37, the norm is given $2/\sqrt{3}$ and bounded by 1.72, respectively.

Remark 1. Our implementation covers cases of $s = 1$ and $p = 17, 257$. In these cases, c^2 is approximately $2.06, 2.01$, respectively.

Remark 2. Since $\|a\|_2 \leq \|a\|_\infty$, the distinguishing problem given by ℓ_∞ norm is at least as hard as the problem given by ℓ_2 norm. In other words, m-PLWE sample can be chosen by an error distribution following ℓ_∞ norm rather than ℓ_2 norm. From now on, the norm of m-PLWE samples, or their errors, are measured by ℓ_∞ norm.

2.4 HEAAN Scheme

The following is the instantiation of the RLWE-based `HEAAN` scheme [11, 12]. For a power-of-two $N > 4$ and $M = 2N$, denote $\Phi_M(x) = (x^N + 1)$, $\mathcal{R} = \mathbb{Z}[x]/\Phi_M(x)$. For a positive integer ℓ, denote $\mathcal{R}_\ell = \mathcal{R}/2^\ell\mathcal{R} = \mathbb{Z}_{2^\ell}[x]/\Phi_M(x)$ the residue ring of $\mathcal{R}$ modulo 2^ℓ. The variant of the canonical embedding map defined as

$$\tau'_{N/2} : m(x) \to \mathbf{z} = (z_0, \ldots, z_{N/2-1})$$

such that $z_j = m(\xi_M^{5^j})$.

Sparse packing. For a power-of-two $n \leq N/2$ consider a subring $\mathcal{R}^{(n)} = \mathbb{Z}[x']/(x'^{2n} + 1) \subset \mathcal{R}$ where $x' = x^{N/(2n)}$. For $\mathcal{R}^{(n)}$ define an isomorphism $\tau'_n : m(x') = m(x^{N/(2n)}) \to \mathbf{z} = (z_0, \ldots, z_{n-1})$ such that $z_j = m(\xi'_j)$, where

$\xi'_j = \xi_j^{N/(2n)}$. We can pack n complex values via isomorphism ${\tau'}_n^{-1}$. In this case if we apply ${\tau'}_{N/2}$ to $m(x') \in \mathcal{R}$ we will get a vector obtained from $\mathbf{z}$ by concatenating itself N/n times. For a message $m(x)$ encoding a vector $\mathbf{z}$ and a ciphertext ct encrypting $m(x)$, ct is also said to be encrypting vector $\mathbf{z}$.

- $\underline{\texttt{HEAAN.KeyGen}(1^\lambda)}$.
  - For an integer L that corresponds to the largest ciphertext modulus level, 2^{2L} is the largest modulus, which corresponds to switching keys. Given the security parameter λ, output the ring dimension N which is a power of two.
  - Set the small distributions $\chi_{key}, \chi_{err}, \chi_{enc}$ over $\mathcal{R}$ for secret, error, and encryption, respectively.
  - Sample a secret $s \leftarrow \chi_{key}$, a random $a \leftarrow \mathcal{R}_L$ and an error $e \leftarrow \chi_{err}$. Set the secret key as $\mathsf{sk} \leftarrow (s, 1)$ and the public key as $\mathsf{pk} \leftarrow (a, b) \in \mathcal{R}_L^2$ where $b \leftarrow -as + e \pmod{2^L}$.
- $\underline{\texttt{HEAAN.KSGen}_{\mathsf{sk}}(s')}$. For $s' \in \mathcal{R}$, sample a random $a' \leftarrow \mathcal{R}_{2\cdot L}$ and an error $e' \leftarrow \chi_{err}$. Output the switching key as $\mathsf{swk} \leftarrow (a', b') \in \mathcal{R}_{2\cdot L}^2$ where $b' \leftarrow -a's + e' + 2^L s' \pmod{2^{2\cdot L}}$.
  - Set the evaluation key as $\mathsf{evk} \leftarrow \texttt{HEAAN.KSGen}_{\mathsf{sk}}(s^2)$.
- $\underline{\texttt{HEAAN.Encode}(\mathbf{z}, p)}$. For a vector $\mathbf{z} \in \mathbb{C}^n$, with of a power-of-two $n \le N/2$ and an integer $p < L$ corresponding to precision bits, output the polynomial $m \leftarrow {\tau'}_n^{-1}(2^p \cdot \mathbf{z}) \in \mathcal{R}$.
- $\underline{\texttt{HEAAN.Decode}(m, p)}$. For a plaintext $m \in \mathcal{R}$, the encoding of a vector consisting of a power-of-two $n \le N/2$ complex messages and precision bits p, output the vector $\mathbf{z} \leftarrow {\tau'}_n(m/2^p) \in \mathbb{C}^n$.
- $\underline{\texttt{HEAAN.Enc}_{\mathsf{pk}}(m)}$. For $m \in \mathcal{R}$, sample $v \leftarrow \chi_{enc}$ and $e_0, e_1 \leftarrow \chi_{err}$. Output $v \cdot \mathsf{pk} + (e_0, e_1 + m) \pmod{2^L}$.
- $\underline{\texttt{HEAAN.Dec}_{\mathsf{sk}}(\mathsf{ct})}$. For $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_\ell^2$, output $c_0 \cdot s + c_1 \pmod{2^\ell}$.
- $\underline{\texttt{HEAAN.Add}(\mathsf{ct}_1, \mathsf{ct}_2)}$. For $\mathsf{ct}_1, \mathsf{ct}_2 \in \mathcal{R}_\ell^2$, output $\mathsf{ct}_{\mathsf{add}} \leftarrow \mathsf{ct}_1 + \mathsf{ct}_2 \pmod{2^\ell}$.
- $\underline{\texttt{HEAAN.CMult}_{\mathsf{evk}}(\mathsf{ct}, \mathbf{c}, p)}$. For $\mathsf{ct} \in \mathcal{R}_\ell^2$ and $\mathbf{c} \in \mathbb{C}^n$, compute $c \leftarrow \texttt{HEAAN.Encode}(\mathbf{c}; p)$ and output $\mathsf{ct}' \leftarrow c \cdot \mathsf{ct} \pmod{2^\ell}$.
- $\underline{\texttt{HEAAN.PolyMult}_{\mathsf{evk}}(\mathsf{ct}, g, p)}$. For $\mathsf{ct} \in \mathcal{R}_\ell^2$ and $g \in \mathcal{R}_\ell$, output $\mathsf{ct}' \leftarrow g \cdot \mathsf{ct} \pmod{2^\ell}$.
- $\underline{\texttt{HEAAN.Mult}_{\mathsf{evk}}(\mathsf{ct}_1, \mathsf{ct}_2)}$. For $\mathsf{ct}_1 = (a_1, b_1), \mathsf{ct}_2 = (a_2, b_2) \in \mathcal{R}_\ell^2$, let $(d_0, d_1, d_2) = (a_1 \cdot a_2, a_1 \cdot b_2 + a_2 \cdot b_1, b_1 \cdot b_2) \pmod{2^\ell}$. Output $\mathsf{ct}_{\mathsf{mult}} \leftarrow (d_1, d_2) + \lfloor 2^{-L} \cdot d_0 \cdot \mathsf{evk} \rceil \pmod{2^\ell}$.
- $\underline{\texttt{HEAAN.ReScale}(\mathsf{ct}, p)}$. For a ciphertext $\mathsf{ct} \in \mathcal{R}_\ell^2$ and an integer p, output $\mathsf{ct}' \leftarrow \lfloor 2^{-p} \cdot \mathsf{ct} \rceil \pmod{2^{\ell - p}}$.
- $\underline{\texttt{HEAAN.ModDown}(\mathsf{ct}, p)}$. For a ciphertext $\mathsf{ct} \in \mathcal{R}_\ell^2$ and an integer p, output $\mathsf{ct}' \leftarrow \mathsf{ct} \pmod{2^{\ell - p}}$.

For an integer k co-prime with M, let $\kappa_k : m(x) \to m(x^k) \pmod{\Phi_M(x)}$. This transformation can be used to provide more functionalities on plaintext slots.

- $\underline{\texttt{HEAAN.Conjugate}_{\mathsf{cjk}}(\mathsf{ct})}$. Set the conjugation key as $\mathsf{cjk} \leftarrow \texttt{HEAAN.KSGen}_{\mathsf{sk}}(\kappa_{-1}(s))$. For $\mathsf{ct} = (a, b) \in \mathcal{R}_\ell^2$ encrypting vector $\mathbf{z}$, let $(a', b') = (\kappa_{-1}(a), \kappa_{-1}(b)) \pmod{2^\ell}$. Output $\mathsf{ct}_{\mathsf{cj}} \leftarrow (0, b') + \lfloor 2^{-L} \cdot a' \cdot \mathsf{cjk} \rceil \pmod{2^\ell}$. $\mathsf{ct}_{\mathsf{cj}}$ is a ciphertext encrypting $\bar{\mathbf{z}}$ - the conjugated plaintext vector of ct.
- $\underline{\texttt{HEAAN.Rotate}_{\mathsf{rtk}}(\mathsf{ct}; r)}$. Set the rotation key as $\mathsf{rtk} \leftarrow \texttt{HEAAN.KSGen}_{\mathsf{sk}}(\kappa_{5^r}(s))$. For $\mathsf{ct} = (a, b) \in \mathcal{R}_\ell^2$ encrypting vector $\mathbf{z}$, let $(a', b') = (\kappa_{5^r}(a), \kappa_{5^r}(b)) \pmod{2^\ell}$. Output $\mathsf{ct}_{\mathsf{rt}} \leftarrow (0, b') + \lfloor 2^{-L} \cdot a' \cdot \mathsf{rtk} \rceil \pmod{2^\ell}$. $\mathsf{ct}_{\mathsf{rt}}$ is a ciphertext encrypting $\mathsf{rt}(\mathbf{z}, r) = (z_r, \ldots, z_{n-1}, z_0, \ldots, z_{r-1})$ - rotated by r positions plaintext vector of ct.

Refer [12, 10] for the technical details and noise analysis.

2.5 Bootstrapping for HEAAN

Consider a ciphertext $\mathsf{ct} \in {\mathcal{R}'}_\ell^2$, an encryption of message $m(x)$ encoding a vector of size n. Then the coefficients of $m(x)$ are non-zero only at degrees $k \cdot \frac{N}{2n}$ for $k = 1, 2, \cdots, 2n-1$. Consider ct as an element of ${\mathcal{R}'}_L^2$ for $L \gg \ell$. We can treat ct as an encryption of $m(x) + 2^\ell \cdot I(x)$ in ${\mathcal{R}'}_L$ i.e. $\texttt{Dec}(\mathsf{ct}) = m(x) + e(x) + 2^\ell \cdot I(x) \pmod{\mathcal{R}}$ for some polynomial $I(x)$ of degree $< N$. With a choice of sparse sk, coefficients of $I(x)$ are bounded with some constant. Now the bootstrapping procedure is defined as followings.

- $\underline{\texttt{HEAAN.SubSum}(\mathsf{ct}, n)}$ As the number of slots is n, then nonzero coefficients of $m(x)$ are only at degrees $k \cdot \frac{N}{2n}$. The output encrypts a message $m(x) + 2^\ell \cdot I'(x)$ where $I'(x)$ derived from $I(x)$ by vanishing the coefficients at degrees other than multiples of $\frac{N}{2n}$.

Algorithm 1 SubSum procedure

```
1: procedure SubSum($\mathsf{ct} \in {\mathcal{R}'}_L^2, n \mid N/2, n \geq 1$)
2:     $\mathsf{ct}' \leftarrow \mathsf{ct}$
3:     for $j = 0$ to $\log(\frac{N}{2n}) - 1$ do
4:         $\mathsf{ct}_j \leftarrow$ HEAAN.Rotate($\mathsf{ct}'; 2^j \cdot n$)
5:         $\mathsf{ct}' \leftarrow$ HEAAN.Add($\mathsf{ct}', \mathsf{ct}_j$)
6:     end for
7:     $\mathsf{ct}'' \leftarrow$ HEAAN.ReScale($\mathsf{ct}'; \log(\frac{N}{2n})$)
8:     return $\mathsf{ct}''$
9: end procedure
```

Let $m(x) + 2^\ell \cdot I'(x) = \sum_{j=0}^{N-1} t_j x^j$ encoding vector $\mathbf{z} = (z_0, \ldots, z_{n-1})$. Then for the following matrix Σ we have equation:

$$\Sigma \cdot \mathbf{z} = \begin{bmatrix} \xi_0^0 & \xi_0^1 & \cdots & \xi_0^{n-1} \\ \xi_1^0 & \xi_1^1 & \cdots & \xi_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{n-1}^0 & \xi_{n-1}^1 & \cdots & \xi_{n-1}^{n-1} \end{bmatrix} \cdot \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} t'_0 + i \cdot t'_n \\ t'_1 + i \cdot t'_{n+1} \\ \vdots \\ t'_{n-1} + i \cdot t'_{2n-1} \end{bmatrix} \tag{1}$$

where $\xi_j = \exp(\frac{2\pi i \cdot 5^j}{2n})$ and $t'_k = t_{k \cdot \frac{N}{2n}}$.

- $\underline{\texttt{HEAAN.SlotToCoeff(ct)}}$. Multiply ct by a matrix Σ^{-1}. The output is the ciphertext that encrypts coefficients of $m(x) + 2^\ell \cdot I'(x)$ in real and imaginary parts: $t_{k \cdot \frac{N}{2n}} + i \cdot t_{(k+n) \cdot \frac{N}{2n}}$ in slot k for $k = 1, 2, \cdots, n-1$.
- $\underline{\texttt{HEAAN.RemoveIPart(ct)}}$ Extract real and imaginary parts of slots and evaluate the polynomial function, close to $f(x) = \frac{1}{2\pi i} \exp(\frac{2\pi i x}{2^\ell})$ for both parts. Combine the two ciphertexts to obtain a ciphertext that encrypts coefficients of $m(x)$ in real and imaginary parts: $m_{k \cdot \frac{N}{2n}} + i \cdot m_{(k+n) \cdot \frac{N}{2n}}$ in slot k for $k = 1, 2, \cdots, n-1$.
- $\underline{\texttt{HEAAN.CoeffToSlot(ct)}}$ Multiply ct by a matrix Σ^{-1}. The result is a ciphertext that encrypts $m(x)$ in a higher power-of-two modulus $L' \gg \ell$

`SlotToCoeff` and `CoeffToSlot` parts of the algorithm require $O(\sqrt{n})$ ciphertext rotations and $O(n)$ constant multiplications when performing so-called 'baby-giant step' optimization. The algorithm also requires to store $O(\sqrt{n})$ rotations keys, which is impractical for large number of slots. For more details refer to [10, 11].

3 MHEAAN Scheme

3.1 Structure of MHEAAN

In this section we will use notations from Section 2.2. MHEAAN is a generalization of HEAAN to a case of non power-of-two M. The encryption process in MHEAAN scheme can be shown in the following outline: we encode a tensor of complex values of size $\hat{\mathbf{N}}$ using ${\tau'}^{-1}_{\mathbf{N}_h}$ into $m(\mathbf{x}) \in \mathcal{R}'$. We mask the result with m-PLWE instance $\big(a(\mathbf{x}), b(\mathbf{x})\big)$ in the corresponding ring $\mathcal{R}'_\ell$. For a message $m(\mathbf{x})$ encoding a tensor $\hat{\mathbf{z}}$ and a ciphertext ct encrypting $m(\mathbf{x})$, we also say that ct encrypts tensor $\hat{\mathbf{z}}$.

Sparse packing. For divisors n_0 of $N_0/2$ and n_i of N_i for $i = 1, \ldots, s$, denote $\hat{\mathbf{n}} = n_0 \times n_1 \times \cdots \times n_s$, $\mathbf{n} = (n_0, n_1, \ldots, n_s)$. We can imitate sparse tensor packing similar to the HEAAN case. We can encode a sparse tensor of complex values of size $\hat{\mathbf{n}}$ using ${\tau'}^{-1}_{\mathbf{N}_h}$ applied to a tensor of size $\hat{\mathbf{N}}_h$ consisting of same blocks of size $\hat{\mathbf{n}}$. We denote this embedding as ${\tau'}^{-1}_{\mathbf{n}}$.

We can treat HEAAN scheme as a special case of MHEAAN with $s = 0$:

$$\mathbf{z} = \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_{n_0-1} \end{bmatrix} \xrightarrow[\texttt{Encode}]{{\tau'}_{n_0}^{-1}} m(x) \xrightarrow[\texttt{Enc}]{\mathsf{RLWE}} \mathsf{ct}$$

and for two-dimensional packing ($s = 1$) we have:

$$\mathbf{Z} = \begin{bmatrix} z_{0,0} & z_{0,1} & \cdots & z_{n_1-1} \\ z_{1,0} & z_{1,1} & \cdots & z_{1,n_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n_0-1,0} & z_{n_0-1,1} & \cdots & z_{n_0-1,n_1-1} \end{bmatrix} \xrightarrow[\texttt{Encode}]{{\tau'}_{n_0,n_1}^{-1}} m(x_0, x_1) \xrightarrow[\texttt{Enc}]{m-\mathsf{RLWE}} \mathsf{ct}$$

3.2 Concrete Construction

For a positive integer ℓ denote $\mathcal{R}'_\ell = \mathcal{R}'/2^\ell\mathcal{R}'$ the residue ring of $\mathcal{R}'$ modulo 2^ℓ. For a real $\sigma > 0$, $\mathcal{DG}(\sigma^2)$ samples a multivariate polynomial in $\mathcal{R}'$ by drawing its coefficient independently from the discrete Gaussian distribution of variance σ^2. For an positive integer h, $\mathcal{HWT}(h)$ is the set of signed binary tensors in $\{0, \pm 1\}^{\hat{\mathbf{N}}}$ whose Hamming weight is exactly h. For a real $0 \le \rho \le 1$, the distribution $\mathcal{ZO}(\rho)$ draws each entry in the tensor from $\{0, \pm 1\}^{\hat{\mathbf{N}}}$, with probability $\rho/2$ for each of -1 and $+1$, and probability being zero $1 - \rho$.

- $\underline{\texttt{MHEAAN.KeyGen}(1^\lambda)}$.
  - Given the security parameter λ, set an integer M that corresponds to a cyclotomic ring, an integer L that corresponds to the largest ciphertext modulus level and distribution parameters (ρ, σ, h).
  - Set the distributions $\chi_{enc} = \mathcal{ZO}(\rho)$, $\chi_{err} = \mathcal{DG}(\sigma)$, $\chi_{key} = \mathcal{HWT}(h)$ over $\mathcal{R}$ for secret, error, and encryption, respectively.
  - Sample a secret $s \leftarrow \chi_{key}$, a random $a \leftarrow {\mathcal{R}'}_L$ and an error $e \leftarrow \chi_{err}$. Set the secret key as $\mathsf{sk} \leftarrow (s, 1)$ and the public key as $\mathsf{pk} \leftarrow (a, b) \in {\mathcal{R}'}_L^2$ where $b \leftarrow -a \cdot s + e \pmod{2^L}$.
- $\underline{\texttt{MHEAAN.KSGen}_{\mathsf{sk}}(s)}$. For $\mathsf{s} \in \mathcal{R}'$, sample a random $a \leftarrow {\mathcal{R}'}_{2\cdot L}$ and an error $e \leftarrow \chi_{err}$. Output the switching key as $\mathsf{swk} \leftarrow (a, b) \in {\mathcal{R}'}_{2\cdot L}^2$ where $b \leftarrow -a \cdot s + e + 2^L s \pmod{{\mathcal{R}'}_{2\cdot L}}$.
  - Set the evaluation key as $\mathsf{evk} \leftarrow \texttt{MHEAAN.KSGen}_{\mathsf{sk}}(s^2)$.
- $\underline{\texttt{MHEAAN.Encode}(\hat{\mathbf{z}}, p)}$. For a tensor $\hat{\mathbf{z}} \in \mathbb{C}^{\hat{\mathbf{n}}}$, an integer $p < L-1$ corresponding to precision bits, output the two-degree polynomial $m \leftarrow {\tau'}_{\mathbf{n}}(2^p \cdot \hat{\mathbf{z}}) \in \mathcal{R}'$.
- $\underline{\texttt{MHEAAN.Decode}(m, p)}$. For a plaintext $m \in \mathcal{R}'$, the encoding of a tensor of complex messages $\hat{\mathbf{z}} \in \mathbb{C}^{\hat{\mathbf{n}}}$, precision bits p, output the tensor $\hat{\mathbf{z}}' \leftarrow {\tau'}_{\mathbf{n}}^{-1}(m/2^p) \in \mathbb{C}^{\hat{\mathbf{n}}}$.
- $\underline{\texttt{MHEAAN.Enc}_{\mathsf{pk}}(m)}$. For $m \in \mathcal{R}'$, sample $v \leftarrow \chi_{enc}$ and $e_0, e_1 \leftarrow \chi_{err}$. Output $\mathsf{ct} = v \cdot \mathsf{pk} + (e_0, e_1 + m) \pmod{{\mathcal{R}'}_L}$.

- $\underline{\texttt{MHEAAN.Dec}_{\mathsf{sk}}(\mathsf{ct})}$. For $\mathsf{ct} = (c_0, c_1) \in {\mathcal{R}'}^2_\ell$, output $c_0 \cdot s + c_1 \pmod{\mathcal{R}'_\ell}$.
- $\underline{\texttt{MHEAAN.Add}(\mathsf{ct}_1, \mathsf{ct}_2)}$. For $\mathsf{ct}_1, \mathsf{ct}_2 \in {\mathcal{R}'}^2_\ell$ - encryption of tensors $\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2 \in \mathbb{C}^{\hat{\mathbf{n}}}$ output $\mathsf{ct}_{\mathsf{add}} \leftarrow \mathsf{ct}_1 + \mathsf{ct}_2 \pmod{2^\ell}$. $\mathsf{ct}_{\mathsf{add}}$ is a ciphertext encrypting tensor $\hat{\mathbf{z}}_1 + \hat{\mathbf{z}}_2$.
- $\underline{\texttt{MHEAAN.CMult}_{\mathsf{evk}}(\mathsf{ct}, \mathbf{C}, p)}$. For $\mathsf{ct} \in \mathcal{R}^2_\ell$ - encryption of $\hat{\mathbf{z}} \in \mathbb{C}^{\hat{\mathbf{n}}}$, and a constant tensor $\hat{\mathbf{c}} \in \mathbb{C}^{\hat{\mathbf{n}}}$, compute $c \leftarrow \texttt{MHEAAN.Encode}(\hat{\mathbf{c}}, p)$ the encoding of $\hat{\mathbf{c}}$ and output $\mathsf{ct}_{\mathsf{cmult}} \leftarrow c \cdot \mathsf{ct} \pmod{\mathcal{R}'_\ell}$. $\mathsf{ct}_{\mathsf{cmult}}$ is a ciphertext encrypting tensor $\hat{\mathbf{z}} \odot \hat{\mathbf{c}}$.
- $\underline{\texttt{MHEAAN.PolyMult}_{\mathsf{evk}}(\mathsf{ct}, g, p)}$. For $\mathsf{ct} \in \mathcal{R}^2_\ell$ - encryption of $\hat{\mathbf{z}} \in \mathbb{C}^{\hat{\mathbf{n}}}$, and a constant $g \in \mathcal{R}_\ell$ output $\mathsf{ct}_{\mathsf{cmult}} \leftarrow c \cdot \mathsf{ct} \pmod{\mathcal{R}'_\ell}$. $\mathsf{ct}_{\mathsf{cmult}}$ is a ciphertext encrypting tensor $\hat{\mathbf{z}} \odot \hat{\mathbf{c}}$, where $\hat{\mathbf{c}}$ is decoding of g.

Multiplication by polynomial is similar to a constant multiplication, however in the next section we will show why it is important to define it separately.

- $\underline{\texttt{MHEAAN.Mult}_{\mathsf{evk}}(\mathsf{ct}_1, \mathsf{ct}_2)}$. For $\mathsf{ct}_1 = (a_1, b_1), \mathsf{ct}_2 = (a_2, b_2) \in {\mathcal{R}'}^2_\ell$ - encryptions of tensors $\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2 \in \mathbb{C}^{\hat{\mathbf{n}}}$, let $(d_0, d_1, d_2) = (a_1 a_2,\ a_1 b_2 + a_2 b_1, b_1 b_2) \pmod{\mathcal{R}'_\ell}$. Output $\mathsf{ct}_{\mathsf{mult}} \leftarrow (d_1, d_2) + \lfloor 2^{-L} \cdot d_0 \cdot \mathsf{evk} \rceil \pmod{\mathcal{R}'_\ell}$. $\mathsf{ct}_{\mathsf{mult}}$ is a ciphertext encrypting tensor $\hat{\mathbf{z}}_1 \odot \hat{\mathbf{z}}_2$.
- $\underline{\texttt{MHEAAN.ReScale}(\mathsf{ct}, p)}$. For a ciphertext $\mathsf{ct} \in {\mathcal{R}'}^2_\ell$ and an integer p, output $\mathsf{ct}' \leftarrow \lfloor 2^{-p} \cdot \mathsf{ct} \rceil \pmod{\mathcal{R}'_{\ell-p}}$.

For an integer vector $\mathbf{k} = (k_0, \ldots, k_s)$ with k_i co-prime with M_i, let

$$\kappa_{\mathbf{k}} : m'(\mathbf{x}) \to m'(\mathbf{x}^{\mathbf{k}}) \pmod{\mathcal{R}'_\ell}$$

This transformation can be used to provide conjugation and rotations in different dimensions on the plaintext matrix.

- $\underline{\texttt{MHEAAN.Conjugate}_{\mathsf{cjk}}(\mathsf{ct})}$. Set the conjugation key as

$$\mathsf{cjk} \leftarrow \texttt{MHEAAN.KSGen}_{\mathsf{sk}}(\kappa_{-\mathbf{1}}(s))$$

  For $\mathsf{ct} = (a, b) \in {\mathcal{R}'}^2_\ell$ encrypting matrix $\mathbf{Z}$, let $(a', b') = (\kappa_{-\mathbf{1}}(a), \kappa_{-\mathbf{1}}(b)) \pmod{\mathcal{R}'_\ell}$. Output $\mathsf{ct}_{\mathsf{cj}} \leftarrow (0, b') + \lfloor 2^{-L} \cdot a' \cdot \mathsf{cjk} \rceil \pmod{\mathcal{R}'_\ell}$. $\mathsf{ct}_{\mathsf{cj}}$ is a ciphertext encrypting $\bar{\hat{\mathbf{z}}}$ - the conjugated plaintext tensor of ct.
- $\underline{\texttt{MHEAAN.Rotate}_{\mathsf{rtk}}(\mathsf{ct}; \mathbf{r})}$. Set the rotation key as

$$\mathsf{rtk} \leftarrow \texttt{MHEAAN.KSGen}_{\mathsf{sk}}(\kappa_{\mathbf{g}^{\mathbf{r}}}(s))$$

  For $\mathsf{ct} = (a, b) \in {\mathcal{R}'}^2_\ell$ encrypting matrix $\mathbf{Z}$, let $(a', b') = (\kappa_{\mathbf{g}^{\mathbf{r}}}(a), \kappa_{\mathbf{g}^{\mathbf{r}}}(b)) \pmod{\mathcal{R}'_\ell}$. Output $\mathsf{ct}_{\mathsf{rt}} \leftarrow (0, b') + \lfloor 2^{-L} \cdot a' \cdot \mathsf{rtk} \rceil \pmod{\mathcal{R}'_\ell}$. $\mathsf{ct}_{\mathsf{rt}}$ is a ciphertext encrypting $\mathsf{rt}(\hat{\mathbf{z}}, \mathbf{r})$ - cyclic rotated plaintext tensor by r_i in i-th dimension.

Throughout this paper, we use real polynomials as plaintexts for convenience of analysis. A ciphertext $\mathsf{ct} \in {\mathcal{R}'}_\ell^2$ will be called a valid encryption of $m \in \mathcal{S}$ with the encryption noise bounded by δ, and plaintext bounded by μ, if $\langle \mathsf{ct}, \mathsf{sk} \rangle = m + e \pmod{\mathcal{R}'_\ell}$ for some polynomial $e \in \mathcal{S}$ with $\|e\|_\infty^{\mathsf{can}} < \delta$ and $\|m\|_\infty^{\mathsf{can}} < \mu$. We will use a corresponding tuple $(\mathsf{ct}, \delta, \mu, \ell)$ for such an encryption of m. The following lemmas give upper bounds on noise growth after encryption, rescaling and homomorphic operations. Refer to Appendix A for proofs.

Lemma 2 (Encoding & Encryption). *For $m \leftarrow$ MHEAAN.Encode$(\hat{\mathbf{z}}, p)$ and $\mathsf{ct} \leftarrow$ MHEAAN.Enc$_{\mathsf{pk}}(m)$ the encryption noise is bounded by $\delta_{\mathsf{clean}} = 8\sqrt{2} \cdot \sigma N + 6\sigma\sqrt{N} + 16\sigma\sqrt{hN}$.*

Lemma 3 (Rescaling). *Let $(\mathsf{ct}, \delta, \mu, \ell)$ be a valid encryption of m and $\mathsf{ct}' \leftarrow$ MHEAAN.ReScale(ct, p). Then $(\mathsf{ct}', \delta/2^p + \delta_{\mathsf{scale}}, \mu/2^p, \ell - p)$ is a valid encryption of $m/2^p$ where $\delta_{\mathsf{scale}} = 6\sqrt{N/12} + 16\sqrt{hN/12}$*

Remark 3. We can slightly change the public key generation and the encryption process to obtain a ciphertext with initial noise reduced from δ_{clean} to almost δ_{scale}. For this, as 2^{2L} is the largest modulus, which corresponds to switching keys, we can generate public key in ${\mathcal{R}'}_{2L}^2$ instead of ${\mathcal{R}'}_L^2$. Also in the encryption process we encode the plaintext m with $p + L$ precision bits, instead of p bits with the following rescaling of the encryption ct of m by L bits. With a slightly slower encryption process we end up with a valid encryption in ${\mathcal{R}'}_L^2$, with the initial noise bounded by $\delta_{\mathsf{clean}}/2^L + \delta_{\mathsf{scale}} \approx \delta_{\mathsf{scale}}$.

Lemma 4 (Addition & Multiplication). *Let $(\mathsf{ct}_i, \delta_i, \mu_i, \ell)$ be encryptions of $m_i \in \mathcal{R}'$ and let $\mathsf{ct}_{\mathsf{add}} \leftarrow$ MHEAAN.Add$(\mathsf{ct}_1, \mathsf{ct}_2)$ and $\mathsf{ct}_{\mathsf{mult}} \leftarrow$ MHEAAN.Mult$_{\mathsf{evk}}(\mathsf{ct}_1, \mathsf{ct}_2)$. Then $(\mathsf{ct}_{\mathsf{add}}, \delta_1 + \delta_2, \mu_1 + \mu_2, \ell)$ and $(\mathsf{ct}_{\mathsf{mult}}, \mu_1 \cdot \delta_2 + \mu_2 \cdot \delta_1 + \delta_1 \cdot \delta_2 + \delta_{\mathsf{mult}}, \mu_1 \cdot \mu_2, \ell)$ are valid encryptions of $m_1 + m_2$ and $m_1 \cdot m_2$, respectively, where $\delta_{\mathsf{ks}} = 8\sigma N/\sqrt{3}$ and $\delta_{\mathsf{mult}} = 2^{\ell - L} \cdot \delta_{\mathsf{ks}} + \delta_{\mathsf{scale}}$.*

Lemma 5 (Conjugation & Rotation). *Let $(\mathsf{ct}, \delta, \mu, \ell)$ be encryption of $m \in \mathcal{R}'$ that encodes tensor $\hat{\mathbf{z}}$, $\mathbf{r}$- integer vector, and let $\mathsf{ct}_{\mathsf{rt}} =$ MHEAAN.Rotate$_{\mathsf{rtk}}(\mathsf{ct}; \mathbf{r})$, $\mathsf{ct}_{\mathsf{cj}} =$ MHEAAN.Conjugate$_{\mathsf{cjk}}(\mathsf{ct})$. Then $(\mathsf{ct}_{\mathsf{rt}}, \delta + \delta_*, \mu, \ell)$ and $(\mathsf{ct}_{\mathsf{cj}}, \delta + \delta_*, \mu, \ell)$ are valid encryptions of tensors $\mathsf{rt}(\hat{\mathbf{z}}, \mathbf{r})$ and $\bar{\hat{\mathbf{z}}}$ respectively where where $\delta_{\mathsf{ks}} = 8\sigma N/\sqrt{3}$ and $\delta_* = 2^{\ell - L} \cdot \delta_{\mathsf{ks}} + \delta_{\mathsf{scale}}$*

Relative Error As discussed in [12] the decryption of a ciphertext is an approximate value of plaintext, so it needs to dynamically manage the bound of noise of ciphertext. It is sometimes convenient to consider the relative error defined by $\beta = \delta/\mu$. When two ciphertexts with relative errors $\beta_i = \delta_i/\mu_i$ are added the output ciphertext has a relative error bounded by $\max_i(\beta_i)$. When two ciphertexts are multiplied with the following rescaling by p bits the output ciphertext has a relative error bounded by

$$\beta' = \beta_1 + \beta_2 + \beta_1\beta_2 + \frac{\delta_{\mathsf{mult}} + 2^p \cdot \delta_{\mathsf{scale}}}{\mu_1\mu_2}$$

according to Lemmas 3 and 4. This relative error is close to $\beta_1 + \beta_2$ which is similar to the case of unencrypted floating-point multiplication under an appropriate choice of parameters.

For convenience of analysis, we will assume that for two ciphertexts with relatives errors β_1 and β_2 the relative error after multiplication and rescaling is bounded by $\beta_1 + \beta_2 + \beta^*$ for some fixed β^*

4 Bootstrapping for MHEAAN

Similar to HEAAN scheme, consider a ciphertext $\mathtt{ct} \in {\mathcal{R}'}_\ell^2$ as an element of ${\mathcal{R}'}_L^2$ for $L \gg \ell$, with $\mathtt{Dec}(\mathtt{ct}) = m(\mathbf{x}) + e(\mathbf{x}) + 2^\ell \cdot I(\mathbf{x}) \pmod{{\mathcal{R}'}_L}$. For simplicity we only consider boostrapping for full packing. However some cases of sparse packing (as sparse packing in dimension corresponding to M_0) could be achieved using similar to HEAAN case techniques.

- <u>MHEAAN.SlotToCoeff(ct)</u>. From the equation 2 (in appendix) we notice that linear transformation can be split into consecutive linear transformations consisting of Σ from the equation 1 and Σ'_i from the equations 3 applying to different dimensions i of $m(\mathbf{x})$. Output is the ciphertext that encrypts coefficients of $m(\mathbf{x}) + e(\mathbf{x}) + 2^\ell \cdot I(\mathbf{x})$ in real and imaginary parts.
- <u>MHEAAN.RemoveIPart(ct)</u> This part of algorithm is same to HEAAN. Extract real and imaginary parts of slots, evaluate polynomial function, close to $f(x) = \frac{1}{2\pi i}\exp(\frac{2\pi i x}{2^\ell})$ for both parts. Combine two ciphertexts to obtain ciphertext that encrypts coefficients of $m(\mathbf{x})$ in real and imaginary parts.
- <u>HEAAN.CoeffToSlot(ct)</u> Apply consecutively linear transformations Σ^{-1} and Σ_i^{-1}. The result is a ciphertext that encrypts same vector as initial ct in a higher modulus ${\mathcal{R}'}_{L'}^2$ with $L' \gg \ell$.

The noise, correctness and performance analysis are similar to [10] with the differences that now SlotToCoeff and CoeffToSlot parts of the algorithm require $O(\sum_{i=0}^s \sqrt{N_i})$ ciphertext rotations and $O(\sum_{i=0}^s N_i)$ constant multiplications when performing 'baby-giant step' optimization. This is much smaller than $O(\sqrt{N})$ and $O(N)$ corresponding to HEAAN case for a full slot packing $N/2$. We now also have to store only $O(\sum_{i=0}^s \sqrt{N_i})$ rotations keys instead of $O(\sqrt{N})$ keys for HEAAN case. The only drawback is that when applying consecutively linear transformations, we use more rescaling operations. For small s such as $s = 1$, however, it is not a big issue.

5 Homomorphic Evaluations of Matrix Operations

One of the purposes to design MHEAAN is to run the matrix operations naturally. Since a matrix multiplication consists of multiplications and additions for each components, every HE scheme should support the operation. However, the there is no known general practical result yet. With the structure of MHEAAN we provide algorithms for homomorphic evaluation of approximate matrix multiplication, transposition and inverse functions.

Let n be a divisor of both of $N_0/2$ and N_1, in particular n is a power-of-two. For simplicity we will consider only square power-of-two size matrix case for multiplication, transposition and inverse. One can keep in mind parameters $(s, M_0, M_1) = (1, 2^k, 257)$, in which case n can be up to $min(2^{k-2}, 256)$, and parameters $(s, M_0, M_1) = (1, 2^k, 17)$, in which case n can be up to $min(2^{k-2}, 16)$. We start with several simple auxiliary algorithms.

Remark 4. Multiplication and transposition algorithms can be extended to a non-square matrices case. Also for bigger matrices we can split them into smaller ones and use divide-and-conquer algorithm. We will omit the details as we need to consider many cases, although they are essentially similar.

Row and Column Sums Let $\mathsf{ct}_\mathbf{A}$ - encryption of matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$. Then the algorithm 2 return the ciphertext encrypting row sums of $\mathbf{A}$. Similarly we can define algorithm `ColSum` for column sums of $\mathbf{A}$.

Algorithm 2 Row Sum

```
1: procedure MHEAAN.RowSum($\mathsf{ct}_\mathbf{A} \in \mathcal{R}'^2_\ell$)
2:     $\mathsf{ct}_\mathbf{S} \leftarrow \mathsf{ct}_\mathbf{A}$
3:     for $j = 0$ to $\log n$ do
4:         $\mathsf{ct}_j \leftarrow$ MHEAAN.Rotate($\mathsf{ct}_\mathbf{S}, 2^j, 0$)
5:         $\mathsf{ct}_\mathbf{S} \leftarrow$ MHEAAN.Add($\mathsf{ct}_\mathbf{S}, \mathsf{ct}_j$)
6:     end for
7:     return $\mathsf{ct}_\mathbf{S}$
8: end procedure
```

Diagonal Extraction Let $\mathbf{I} \in \mathbb{C}^{n\times n}$ be the identity matrix with $\mathbf{I}_k = \mathsf{rt}(\mathbf{I}, (k, 0))$. We can obtain encryption of shifted diagonal of $\mathbf{A}$ by multiplying $\mathsf{ct}_\mathbf{A}$ with $\mathbf{I}_k$. The procedure is described in Algorithm 3.

Algorithm 3 Diagonal Extraction

```
1: procedure MHEAAN.Diag($\mathsf{ct}_\mathbf{A} \in \mathcal{R}'^2_\ell, k, p$)
2:     $\mathsf{ct}_{\mathbf{A}_k} \leftarrow$ MHEAAN.CMult($\mathsf{ct}_\mathbf{A}, \mathbf{I}_k$)
3:     $\mathsf{ct}_{\mathbf{A}_k} \leftarrow$ MHEAAN.ReScale($\mathsf{ct}_{\mathbf{A}_k}, p$)
4:     return $\mathsf{ct}_{\mathbf{A}_k}$
5: end procedure
```

5.1 Matrix by Vector Multiplication

Let ciphertext $\mathtt{ct}_{\mathbf{v}}$ encrypts vector $\mathbf{v}$ as a matrix of size $n \times 1$. Remind that $\mathtt{ct}_{\mathbf{v}}$ can be viewed as encryption of matrix of size $n \times n$, consisting of same columns $\mathbf{v}$. If we multiply $\mathtt{ct}_{\mathbf{A}^T}$ by $\mathtt{ct}_{\mathbf{v}}$ and apply `ColSum` algorithm we obtain ciphertext encrypting $\mathbf{w}^T = (\mathbf{Av})^T$ as a matrix of size $1 \times n$. Matrix by vector multiplication is stated in algorithm 4. Similarly for $\mathbf{w}^T$ of size $n \times 1$ we can define `VecMatMult` algorithm that evaluates encryption of $\mathbf{Aw}$.

Algorithm 4 Matrix by Vector Multiplication

```
1: procedure MHEAAN.MatVecMult($\mathtt{ct}_{\mathbf{A}^T}, \mathtt{ct}_{\mathbf{v}} \in \mathcal{R}'^2_\ell, p \in \mathbb{Z}$)
2:     $\mathtt{ct}_{(\mathbf{Av})^T} \leftarrow$ MHEAAN.Mult($\mathtt{ct}_{\mathbf{A}^T}, \mathtt{ct}_{\mathbf{v}}$)
3:     $\mathtt{ct}_{(\mathbf{Av})^T} \leftarrow$ MHEAAN.ReScale($\mathtt{ct}_{(\mathbf{Av})^T}, p$)
4:     $\mathtt{ct}_{(\mathbf{Av})^T} \leftarrow$ MHEAAN.ColSum($\mathtt{ct}_{(\mathbf{Av})^T}$)
5:     return $\mathtt{ct}_{(\mathbf{Av})^T}$
6: end procedure
```

5.2 Matrix Multiplication

We adapt Fox matrix multiplication algorithm [22] to encrypted matrix multiplication. For $\mathtt{ct}_{\mathbf{A}}, \mathtt{ct}_{\mathbf{B}}$ be encryptions of matrices $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n \times n}$ with power-of-two n we define Algorithm 5.

Algorithm 5 Matrix Multiplication

```
1: procedure MHEAAN.MatMult($\mathtt{ct}_{\mathbf{A}}, \mathtt{ct}_{\mathbf{B}} \in \mathcal{R}'^2_\ell, p$)
2:     $\mathtt{ct}_{\mathbf{C}} \leftarrow 0$
3:     for $k = 0$ to $n - 1$ do
4:         $\mathtt{ct}_{\mathbf{B}_k} \leftarrow$ MHEAAN.Diag$_k$($\mathtt{ct}_{\mathbf{B}}, p$)
5:         for $j = 1$ to $\log(n) - 1$ do
6:             $\mathtt{ct}_{\mathbf{B}_k} \leftarrow$ MHEAAN.Add($\mathtt{ct}_{\mathbf{B}_k}$, MHEAAN.Rotate($\mathtt{ct}_{\mathbf{B}_k}, (0, 2^j)$))
7:         end for
8:         $\mathtt{ct}_{\mathbf{A}_k} \leftarrow$ MHEAAN.ModDown(MHEAAN.Rotate($\mathtt{ct}_{\mathbf{A}}, (\frac{N_0}{2} - k, 0)$), $p$)
9:         $\mathtt{ct}_{\mathbf{C}_k} \leftarrow$ MHEAAN.Mult($\mathtt{ct}_{\mathbf{A}_k}, \mathtt{ct}_{\mathbf{B}_k}$)
10:        $\mathtt{ct}_{\mathbf{C}} \leftarrow$ MHEAAN.Add($\mathtt{ct}_{\mathbf{C}}, \mathtt{ct}_{\mathbf{C}_k}$)
11:    end for
12:    $\mathtt{ct}_{\mathbf{C}} \leftarrow$ MHEAAN.ReScale($\mathtt{ct}_{\mathbf{C}}, p$)
13:    return $\mathtt{ct}_{\mathbf{C}}$
14: end procedure
```

Lemma 6 (Matrix Multiplication). *Let* $(\mathsf{ct}_{\mathbf{A}}, \beta_{\mathbf{A}} \cdot 2^p, 2^p, \ell)$ *and* $(\mathsf{ct}_{\mathbf{B}}, \beta_{\mathbf{B}} \cdot 2^p, 2^p, \ell)$ *be encryptions of matrices* $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n \times n}$ *respectively. The Algorithm 5 outputs* $(\mathsf{ct}_{\mathbf{C}}, \beta_{\mathbf{C}} \cdot n \cdot 2^p, n \cdot 2^p, \ell - 2p)$ *the valid encryption of* $\mathbf{C} = \mathbf{AB}$ *where* $\beta_{\mathbf{C}} = \beta_{\mathbf{A}} + \beta_{\mathbf{B}} + (\log n + 1) \cdot \beta^*$.

Remark 5. The plain matrix multiplication algorithm has complexity $O(n^3)$. The Algorithm 5 requires totally $O(n)$ ciphertext multiplication (each of provides multiplication in parallel of n^2 values) and $O(n \log n)$ ciphertext rotations. This is almost optimal, compare to unencrypted case.

Matrix Multiplications with Permutations We will mention about more efficient algorithm for matrix multiplication. If we consider the following permutations of matrices $\mathbf{B}'$ and $\mathbf{C}''$ of $\mathbf{B}$ and $\mathbf{C} = \mathbf{AB}$ respectively.

$$\mathbf{B}' = \begin{bmatrix} b_{0,0} & b_{1,n-1} & \cdots & b_{n-1,1} \\ b_{0,1} & b_{1,0} & \cdots & b_{n-1,2} \\ \vdots & \ddots & \vdots & \\ b_{0,n-1} & b_{1,n-2} & \cdots & b_{n-1,0} \end{bmatrix}, \mathbf{C}'' = \begin{bmatrix} c_{0,0} & c_{0,n-1} & \cdots & c_{0,1} \\ c_{1,1} & c_{1,0} & \cdots & c_{1,2} \\ \vdots & \ddots & \vdots & \\ c_{n-1,n-1} & c_{n-1,n-2} & \cdots & c_{n-1,0} \end{bmatrix}$$

Then for given encryptions of $\mathbf{A}$ and $\mathbf{B}'$, Algorithm 6 outputs encryption of $\mathbf{C}''$ - permutation of matrix $\mathbf{C}$. The Algorithm 6 requires totally $O(n)$ ciphertext multiplication (each of provides multiplication in parallel of n^2 values) and $O(n)$ ciphertext rotations. This is asymptotically optimal, compare to unencrypted case. However this algorithm is seems to be not practical for more complicated tasks as it does not preserve the matrix structure in slots.

Algorithm 6 Matrix Multiplication with Permutations

```
1: procedure MHEAAN.MatMultPermute($\mathsf{ct}_{\mathbf{A}}, \mathsf{ct}_{\mathbf{B}'} \in \mathcal{R}'^2_{\ell}, p$)
2:     $\mathsf{ct}_{\mathbf{C}''} \leftarrow 0$
3:     for $k = 0$ to $n - 1$ do
4:         $\mathsf{ct}_{\mathbf{A}_k} \leftarrow$ MHEAAN.Rotate($\mathsf{ct}_{\mathbf{A}}, (k, 0)$)
5:         $\mathsf{ct}_{\mathbf{B}'_k} \leftarrow$ MHEAAN.Rotate($\mathsf{ct}_{\mathbf{B}'}, (k, k)$)
6:         $\mathsf{ct}_{\mathbf{C}''_k} \leftarrow$ MHEAAN.Mult($\mathsf{ct}_{\mathbf{A}_k}, \mathsf{ct}_{\mathbf{B}'_k}$)
7:         $\mathsf{ct}_{\mathbf{C}''} \leftarrow$ MHEAAN.Add($\mathsf{ct}_{\mathbf{C}}, \mathsf{ct}_{\mathbf{C}''_k}$)
8:     end for
9:     $\mathsf{ct}_{\mathbf{C}''} \leftarrow$ MHEAAN.ReScale($\mathsf{ct}_{\mathbf{C}''}, p$)
10:    return $\mathsf{ct}_{\mathbf{C}''}$
11: end procedure
```

5.3 Matrix Transposition

With `Diag` algorithm we can extract all the shifted diagonals of matrix $\mathbf{A}$. We can notice that transposed matrix $\mathbf{A}^T$ is actually consist of same shifted diagonals $\mathbf{A}_k$ of matrix $\mathbf{A}$, rotated by $(k, -k)$ slots.

Algorithm 7 Matrix Transposition

```
1: procedure MHEAAN.MatTranspose($\mathsf{ct}_{\mathbf{A}} \in \mathcal{R}'^2_\ell, p$)
2:     $\mathsf{ct}_{\mathbf{A}^T} \leftarrow 0$
3:     for $k = 0$ to $n-1$ do
4:         $\mathsf{ct}_{\mathbf{A}_k} \leftarrow$ MHEAAN.Diag$_k(\mathsf{ct}_{\mathbf{A}}, p)$
5:         $\mathsf{ct}_{\mathbf{A}_k} \leftarrow$ MHEAAN.Rotate$(\mathsf{ct}_{\mathbf{A}_k}, (k, -k))$
6:         $\mathsf{ct}_{\mathbf{A}^T} \leftarrow$ MHEAAN.Add$(\mathsf{ct}_{\mathbf{A}^T}, \mathsf{ct}_{\mathbf{A}_k})$
7:     end for
8:     $\mathsf{ct}_{\mathbf{A}_k} \leftarrow$ MHEAAN.ReScale$(\mathsf{ct}_{\mathbf{A}_k}, p)$
9:     return $\mathsf{ct}_{\mathbf{A}_k}$
10: end procedure
```

Lemma 7 (Matrix Transposition). *Let* $(\mathsf{ct}_{\mathbf{A}}, \beta_{\mathbf{A}} \cdot 2^p, 2^p, \ell)$ *be an encryption of matrix* $\mathbf{A} \in \mathbb{C}^{n\times n}$*. The Algorithm 7 outputs* $(\mathsf{ct}_{\mathbf{A}^T}, \beta_{\mathbf{A}^T} \cdot 2^p, 2^p, \ell - p)$ *the valid encryption of* $\mathbf{A}^T$ *where* $\beta_{\mathbf{A}^T} = \beta_{\mathbf{A}} + \beta^*$*.*

5.4 Matrix Inverse

For matrix inverse we can adapt Schulz algorithm [41] to encrypted approximate inverse circuit. However for `MHEAAN` we use a matrix version algorithm described in [9] and adopted in [12] as it more practical due to power-of-two degrees of matrix in the circuit. The algorithm is described below.

Assume that invertible square matrix $\mathbf{A}$ satisfies $\|\bar{\mathbf{A}}\| \le \epsilon < 1$ for $\bar{\mathbf{A}} = \mathbf{I} - \frac{1}{2^t}\mathbf{A}$, for some $t \ge 0$ then we get

$$\frac{1}{2^t}\mathbf{A}(\mathbf{I} + \bar{\mathbf{A}})(\mathbf{I} + \bar{\mathbf{A}}^2)\dots(\mathbf{I} + \bar{\mathbf{A}}^{2^{r-1}}) = 1 - \bar{\mathbf{A}}^{2^r}$$

We can see that $\|\bar{\mathbf{A}}^{2^r}\| \le \|\bar{\mathbf{A}}\|^{2^r} \le \epsilon^{2^r}$, hence $\frac{1}{2^t}\prod_{j=0}^{r-1}(\mathbf{I} + \bar{\mathbf{A}}^{2^j}) = \mathbf{A}^{-1}(1 - \bar{\mathbf{A}^{2^r}})$ is an approximate inverse of $\mathbf{A}$ for $\epsilon^{2^r} \ll 1$. We will slightly strengthen the condition on ϵ in the following lemma:

Lemma 8 (Matrix Inverse). *Let* $(\mathsf{ct}_{\bar{\mathbf{A}}}, \beta \cdot \epsilon 2^p/n, \epsilon 2^p/n, \ell)$ *be an encryption of matrix* $\bar{\mathbf{A}} \in \mathbb{C}^{n\times n}$*, and* $\|\bar{\mathbf{A}}\| = \|\mathbf{I} - \frac{1}{2^t}\mathbf{A}\| \le \epsilon < \frac{n-1}{n}$ *for some* t*. The Algorithm 8 outputs* $(\mathsf{ct}_{\mathbf{V}_r}, \beta_{\mathbf{V}_r} \cdot n^{1/n}2^{p-t}, n^{1/n}2^{p-t}, \ell - 2pr - t)$ *the valid encryption of* $\mathbf{A}^{-1}$ *where* $\beta_{\mathbf{V}_r} = 2\beta + (r+1)\cdot(1 + \log n)\cdot \beta^*$*. So we have that the output message bound is close to* 2^{p-t} *and error growth linearly in* r*.*

Algorithm 8 Matrix Inverse

```
1: procedure MHEAAN.MatInv($\mathsf{ct}_{\bar{\mathbf{A}}} \in \mathcal{R}'^2_\ell, r, p \in \mathbb{Z}$)
2:     $i = \texttt{MHEAAN.Encode}(\mathbf{I}, p)$
3:     $\mathsf{ct}_{\mathbf{A}_0} \leftarrow \mathsf{ct}_{\bar{\mathbf{A}}}$
4:     $\mathsf{ct}_{\mathbf{V}_0} \leftarrow \texttt{MHEAAN.ModDown}(i + \mathsf{ct}_{\bar{\mathbf{A}}}, p)$
5:     for $j = 0$ to $r - 1$ do
6:         $\mathsf{ct}_{\mathbf{A}_j} \leftarrow \texttt{MHEAAN.ReScale}(\texttt{MHEAAN.MatMult}(\mathsf{ct}_{\mathbf{A}_{j-1}}, \mathsf{ct}_{\mathbf{A}_{j-1}}), p)$
7:         $\mathsf{ct}_{\mathbf{V}_{j+1}} \leftarrow \texttt{MHEAAN.ReScale}(\texttt{MHEAAN.MatMult}(\mathsf{ct}_{\mathbf{V}_j}, i + \mathsf{ct}_{\mathbf{A}_j}), p)$
8:     end for
9:     $\mathsf{ct}_{\mathbf{V}_r} \leftarrow \texttt{MHEAAN.ReScale}(\mathsf{ct}_{\mathbf{V}_r}, t)$
10:    return $\mathsf{ct}_{\mathbf{V}_r}$
11: end procedure
```

6 Implementation Results

In this section, we provide implementation results with concrete parameter setting. Our implementation is based on the NTL C++ library running over GMP. Every experimentation was performed on a machine with an Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz processor with 8 CPUs using a parameter set with 80-bit security level.

Parameters Setting The dimensions of a cyclotomic ring $\mathcal{R}'$ are chosen following the security estimator of Albrecht et al. [1] for the learning with errors problem.

Table 1. Parameter settings for `MHEAAN`

parameter	$N = N_0 \cdot N_1$	σ	h	L_{max}
Set_1	2^{13}	6.4	64	≈ 155
Set_2	2^{14}			≈ 310
Set_3	2^{15}			≈ 620
Set_4	2^{16}			≈ 1240

We use the discrete Gaussian distribution of standard deviation σ to sample error polynomials and set the Hamming weight h in a multivariate representation of a secret key $s(\mathbf{x})$.

We skip the results of the evaluation of component wise operations such as inverse, exponent, sigmoid functions, etc. Please refer to [12] for more details on evaluating these circuits.

Bootstrapping In Table 2, we present the parameter setting and performance results for full slots bootstrapping. Parameters r, p, L_{in} have the same meaning as r, $\log(p)$, $\log(q)$ in [10] and similarly were chosen experimentally based on the bootstrapping error. For sufficiently large number r we maintain the precision of the output plaintext. L_{in} and L_{out} corresponds to the number of modulus bits before and after bootstrapping respectively. The running times are only for ciphertext operations and exclude encryption and decryption procedures.

Table 2. Implementation results for bootstrapping

parameter	N_0	N_1	r	p	L_{in}	L_{max}	L_{out}	precision	time	amor
$Boot_1$	256	256	7	35	40	1240	517	16 bits	2.5min	4.58ms
$Boot_2$			8	43	50	1240	312	20 bits	2.63min	4.83ms

Evaluation of Matrix Circuits In Table 3, we present the parameter setting and performance results for matrix multiplication, matrix 16-th power, and inverse. L_{in} and L_{out} corresponds to the number of modulus bits before and after operations respectively. The running times are only for ciphertext operations and exclude encryption and decryption procedures.

The homomorphic evaluation of the circuit $\mathbf{M}^{16}$ can be evaluated by squaring a matrix 4 times. Computing the matrix inverse homomorphically is done by evaluating a matrix polynomial up to degree 15 as was shown in Algorithm 8.

Table 3. Implementation results for $n \times n$ matrices $\mathbf{M}$, $\mathbf{M}_1$, $\mathbf{M}_2$

Function	n	N_0	N_1	p	L_{in}	L_{out}	time
$\mathbf{M}^T$	16	512	16	30	65	35	0.15s
	16	64	256				0.27s
	64	128	256				1.82s
$\mathbf{M}_1\mathbf{M}_2$	16	512	16		100	40	0.51s
	16	64	256				0.98s
	64	128	256				10.72s
$\mathbf{M}^{16}$	16	1024	16		300	60	6.82s
	16	64	256				17.23s
	64	128	256				87.65s
$\mathbf{M}^{-1}$	16	1024	16		300	60	10.61s
	16	64	256				12.87s
	64	128	256				2.1min

7 Evaluation of Deep Neural Network Classification Algorithm

As an application of `MHEAAN` we propose a DNN classification algorithm with an arbitrary number of layers. Our implementation of algorithm is non-interactive and keep both data and model itself in encrypted state.

DNN model. We briefly describe the flow of DNN classification algorithm. DNN model consists of $l+1$ number of layers. For simplicity we enumerate the layers starting from 0. Each layer contains n_i number of nodes for $i = 0, \dots l$. The layer 0 is input layer, the layer l is output layer, and the others are hidden layers. Each of the hidden layers and the output layer has a corresponding weight matrix $\mathbf{W}_i \in \mathbb{R}^{n_i \times n_{i-1}}$ and a bias vector $\mathbf{b}_i \in \mathbb{R}^{n_i}$. For the input vector $\mathbf{a}_0 \in \mathbb{R}^{n_0}$, we consecutively calculate the linear transformation part $\mathbf{z}_i = \mathbf{W}_i \mathbf{a}_{i-1} + \mathbf{b}_i$, and the activation part $\mathbf{a}_i = g_i(\mathbf{z}_i)$ at the each hidden layer. For the output layer we calculate the linear transformation part $\mathbf{z}_l = \mathbf{W}_l \mathbf{a}_{l-1} + \mathbf{b}_l$ and the index of largest value in $\mathbf{z}_l$ is the classification output.

Algorithm 9 Linear Transformation Column to Row

```
1: procedure MHEAAN.LTColRow($\mathsf{ct}_{\mathbf{a}}, \mathsf{ct}_{\mathbf{W}^T}, \mathsf{ct}_{\mathbf{b}^T} \in \mathcal{R}'^2_\ell, p \in \mathbb{Z}$)
2:     $\mathsf{ct}_{(\mathbf{Wa})^T} \leftarrow$ MHEAAN.VecMatMult($\mathsf{ct}_{\mathbf{a}}, \mathsf{ct}_{\mathbf{W}^T}, p$)
3:     $\mathsf{ct}_{\mathbf{z}^T} \leftarrow$ MHEAAN.Add($\mathsf{ct}_{(\mathbf{Wa})^T}, \mathsf{ct}_{\mathbf{b}^T}$)
4:     return $\mathsf{ct}_{\mathbf{z}}$
5: end procedure
```

Algorithm 10 Linear Transformation Row to Column

```
1: procedure MHEAAN.LTRowCol($\mathsf{ct}_{\mathbf{a}^T}, \mathsf{ct}_{\mathbf{W}}, \mathsf{ct}_{\mathbf{b}} \in \mathcal{R}'^2_\ell, p \in \mathbb{Z}$)
2:     $\mathsf{ct}_{\mathbf{Wa}} \leftarrow$ MHEAAN.MatVecMult($\mathsf{ct}_{\mathbf{a}^T}, \mathsf{ct}_{\mathbf{W}}, p$)
3:     $\mathsf{ct}_{\mathbf{z}} \leftarrow$ MHEAAN.Add($\mathsf{ct}_{\mathbf{Wa}}, \mathsf{ct}_{\mathbf{b}}$)
4:     return $\mathsf{ct}_{\mathbf{z}}$
5: end procedure
```

Our approach. For the linear transformation part we use Algorithms 9 and 10 along with divide-and-conquer algorithm. We use sigmoid as activation function for all hidden layers, and for activation part we use the polynomial approximations of sigmoid function, introduced in [31]. In particular we use the degree 7 least square approximation polynomial $g(x)$ of $\frac{1}{1+e^x}$.

$$g(x) = 0.5 - 1.73496 \cdot \frac{x}{8} + 4.19407 \cdot \left(\frac{x}{8}\right)^3 - 5.43402 \cdot \left(\frac{x}{8}\right)^5 + 2.50739 \cdot \left(\frac{x}{8}\right)^7$$

We first apply `LTColRow` to $\mathtt{ct}_{\mathbf{a}_0}$, $\mathtt{ct}_{\mathbf{W}_1^T}$, $\mathtt{ct}_{\mathbf{b}_1^T}$ and obtain $\mathtt{ct}_{\mathbf{z}_1^T}$. Then we evaluate $\mathtt{ct}_{\mathbf{a}_1^T}$ using polynomial approximation $g(x)$ of sigmoid function. After then we apply `LTRowCol` to $\mathtt{ct}_{a_1^T}$, $\mathtt{ct}_{\mathbf{W}_2}$, $\mathtt{ct}_{\mathbf{b}_2}$ to obtain $\mathtt{ct}_{z_2}$ and etc. For each linear transformation part we lose p modulus bits and for each activation part we lose $3p + 3$ modulus bits. With sufficiently large initial modulus bits L and with the bootstrapping algorithm we can evaluate DNN classification with an arbitrary number of layers.

In Table 4 we present the parameter settings, performances, and accuracy results with one, two and four hidden layers. Our DNN classification algorithm applied to MNIST dataset [32] with sigmoid activation functions. The running times are only for ciphertext operations, and exclude times for encryption and decryption procedures. Accuracy is similar to the accuracy of predictions on unencrypted data, which is about 97.9%.

Table 4. Implementation results for DNN prediction phase

parameter	N_0	N_1	p	L_{in}	L_{out}	l	n_0	$(n_1, \ldots, n_l)$	time
DNN_1	128	256	30	300	147	2	784	(1024,10)	55s
DNN_2			30	400	124	3	784	(1024,256,10)	76s
DNN_3			30	600	78	5	784	(1024,1024,1024,256,10)	3.5min
DNN_4	2048	16	30	300	147	2	784	(1024,10)	59s
DNN_5			30	400	124	3	784	(1024,256,10)	81s
DNN_6			30	600	78	5	784	(1024,1024,1024,256,10)	3.8mins

8 Conclusion

In this work, we present `MHEAAN` - a variant of the `HEAAN` homomorphic encryption scheme. `MHEAAN` takes advantage of `HEAAN` by supporting standard approximate HE operations. With a multi-dimensional packing `MHEAAN` enjoys more functionality like efficient operations on matrices and practical bootstrapping even for large number of slots. As an application of `MHEAAN` we propose a non-interactive deep neural network classification algorithm for deep neural network structure with an arbitrary humber of hidden layers.

One of the future works could be applying `MHEAAN` to classification algorithms for general Neural Network architectures like Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Another interesting problem is to achieve learning phase of the Neural Networks with multiple layer structure. We believe that the idea of multi-dimensional variant could have a great potential for these as well as for other applications requiring computations on matrices and tensors.

9 Acknowledgements

This work was partly supported by Institute for Information communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.B0717-16-0098) and by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) (No.2017R1A5A1015626). The authors would like to thank Duhyeong Kim and Yongha Son for giving valuable comments to the technical part of the paper.

References

1. 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.
2. J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding*, pages 45–64. Springer, 2013.
3. F. Bourse, M. Minelli, M. Minihold, and P. Paillier. Fast homomorphic evaluation of deep discretized neural networks. *IACR Cryptology ePrint Archive*, 2017:1114, 2017.
4. Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology–CRYPTO 2012*, pages 868–886. Springer, 2012.
5. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proc. of ITCS*, pages 309–325. ACM, 2012.
6. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS'11, pages 97–106. IEEE Computer Society, 2011.
7. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In *Advances in Cryptology–CRYPTO 2011*, pages 505–524. Springer, 2011.
8. W. Castryck, I. Iliashenko, and F. Vercauteren. Provably weak instances of ring-lwe revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 147–167. Springer, 2016.
9. G. S. Çetin, Y. Doröz, B. Sunar, and W. J. Martin. An investigation of complex operations with word-size homomorphic encryption. Cryptology ePrint Archive, Report 2015/1195, 2015. `http://eprint.iacr.org/2015/1195`.
10. J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. Bootstrapping for approximate homomorphic encryption. Cryptology ePrint Archive, Report 2018/153, 2018. `https://eprint.iacr.org/2018/153`.
11. J. H. Cheon, A. Kim, M. Kim, and Y. Song. Implementation of HEAAN, 2016. `https://github.com/kimandrik/HEAAN`.
12. J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.
13. J. H. Cheon and D. Stehlé. Fully homomophic encryption over the integers revisited. In *Advances in Cryptology–EUROCRYPT 2015*, pages 513–536. Springer, 2015.

14. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Tfhe: Fast fully homomorphic encryption over the torus. *IACR Cryptology ePrint Archive*, 2018:421, 2018.
15. J.-S. Coron, T. Lepoint, and M. Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *Public-Key Cryptography–PKC 2014*, pages 311–328. Springer, 2014.
16. I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology–CRYPTO 2012*, pages 643–662. Springer, 2012.
17. 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.
18. Y. Doröz, Y. Hu, and B. Sunar. Homomorphic AES evaluation using the modified LTV scheme. *Designs, Codes and Cryptography*, 80(2):333–358, 2016.
19. 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.
20. D. H. Duong, P. K. Mishra, and M. Yasuda. Efficient secure matrix multiplication over lwe-based homomorphic encryption. volume 67, pages 69–83. 2016.
21. Y. Elias, K. E. Lauter, E. Ozman, and K. E. Stange. Provably weak instances of ring-lwe. In *Annual Cryptology Conference*, pages 63–92. Springer, 2015.
22. G. Fox and S. Otto. Matrix algorithms on a hypercube i: Matrix multiplication. *Parallel Computing*, 4:17–31, 1987.
23. C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.
24. 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.
25. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*, pages 75–92. Springer, 2013.
26. S. Halevi and V. Shoup. Algorithms in helib. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 554–571, 2014.
27. S. Halevi and V. Shoup. Bootstrapping for helib. In *Advances in Cryptology–EUROCRYPT 2015*, pages 641–670. Springer, 2015.
28. S. Halevi and V. Shoup. Faster homomorphic linear transformations in helib. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 93–120, 2018.
29. E. Hesamifard, H. Takabi, and M. Ghasemi. Cryptodl: Deep neural networks over encrypted data. *CoRR*, abs/1711.05189, 2017.
30. R. Hiromasa, M. Abe, and T. Okamoto. Packing messages and optimizing bootstrapping in gsw-fhe. In *Public-Key Cryptography – PKC 2015*, pages 699–715. Springer, 2015.
31. A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon. Logistic regression model training based on the approximate homomorphic encryption. *https://eprint.iacr.org/2018/254*.
32. Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

33. A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012*, pages 1219–1234. ACM, 2012.
34. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology–EUROCRYPT 2010*, pages 1–23, 2010.
35. P. K. Mishra, D. Rathee, D. H. Duong, and M. Yasuda. Fast secure matrix multiplications over ring-based homomorphic encryption.
36. A. Pedrouzo-Ulloa, J. R. Troncoso-Pastoriza, and F. Pérez-González. Multivariate lattices for encrypted image processing. In *IEEE ICASSP*. 2015.
37. A. Pedrouzo-Ulloa, J. R. Troncoso-Pastoriza, and F. Pérez-González. On ring learning with errors over the tensor product of number fields. 2016. `https://arxiv.org/abs/1607.05244`.
38. C. Peikert. How (not) to instantiate ring-lwe. In *International Conference on Security and Cryptography for Networks*, pages 411–430. Springer, 2016.
39. R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.
40. H. Saberi Najafi and M. Shams Solary. Computational algorithms for computing the inverse of a square matrix, quasi-inverse of a nonsquare matrix and block matrices. *Appl. Math. Comput.*, 183:539–550, 2006.
41. G. Schulz. Iterative berechnung der reziproken matrix. *Zeitschrift für angewandte Mathematik und Mechanik*, 13:57–59, 1933.

A Proofs

We follow the heuristic approach in [24]. Assume that a polynomial $a(\mathbf{x}) \in \mathcal{R}'$ sampled from one of above distributions, so its nonzero entries are independently and identically distributed. Let $\boldsymbol{\xi} = (\xi_{M_0}, \ldots, \xi_{M_s})$ The value $a(\boldsymbol{\xi})$ can be obtained by consecutively computing N/N_i inner products of vectors of coefficients of a corresponding to a power x_i^j for $j = 0, \ldots, N_i - 1$ by a fixed vector $(1, \xi_{M_i}, \ldots, \xi_{M_i}^{N_i})$ of Euclidean norm $\sqrt{N_i}$. Then $a(\boldsymbol{\xi})$ has variance $V = \sigma^2 \prod_{i=0}^{s} N_i = \sigma^2 N$, where σ^2 is the variance of each coefficient of a. Hence $a(\boldsymbol{\xi})$ has the variances $V_U = 2^{2\ell} N/12$, $V_G = \sigma^2 N$ and $V_Z = \rho N$, when a is sampled from $\mathcal{R}_\ell$, $\mathcal{DG}(\sigma^2)$, $\mathcal{ZO}(\rho)$ respectively. In particular, $a(\boldsymbol{\xi})$ has the variance $V_H = h$ when $a(\mathbf{x})$ is chosen from $\mathcal{HWT}(h)$. Moreover, we can assume that $a(\boldsymbol{\xi})$ is distributed similarly to a Gaussian random variable over complex plane since it is a sum of $\phi_{M_0 \cdots M_s}/2$ independent and identically distributed random complex variables. Every evaluations at roots of unity $(\boldsymbol{\xi})$ share the same variance. Hence, we will use 6σ as a high-probability bound on the canonical embedding norm of $a(\mathbf{x})$ when each coefficient has a variance σ^2. For a multiplication of two independent random variables close to Gaussian distributions with variances σ_1^2 and σ_2^2, we will use $16\sigma_1\sigma_2$ as a high-probability bound.

Proof of Proposition 1

Proof. One of such maps $\mathcal{R}' \to \mathcal{R}$ is given by

$$x_j \mapsto x^{M/M_j} \bmod \Phi_M(x) \text{ for all } j = 0, 1, \cdots, s$$

and it extends to

$$\mathcal{S}' = \mathbb{R} \bigotimes_{\mathbb{Z}} \mathcal{R}' \to \mathcal{S} = \mathbb{R} \bigotimes_{\mathbb{Z}} \mathcal{R}$$

At first we check that this map is well-defined. This means that, for all j, x_j and $x_j + \Phi_{M_j}(x_j)$ have same image in $\mathcal{S}$, or simply $\Phi_{M_j}(x^{M/M_j})$ is divisible by $\Phi_M(x)$. Since

$$\Phi_K(x) = \prod_{(k,K)=1, 1 \le k \le K} (x - \zeta_K^k)$$

for any positive integer K and a primitive K-th root of unity $\zeta_K = e^{2\pi i/K}$, we have the following divisibility

$$\Phi_M(x) = \prod_{(k,M)=1,1\le k\le M} (x - \zeta_M^k) \mid \prod_{(k,M)=1,1\le k\le M} (x^{M/M_j} - \zeta_M^{kM/M_j}) = \left(\Phi_{M_j}(x^{M/M_j})\right)^{M_j}.$$

Note that $x - a$ is always a factor of $(x^* - a^*) = (x-a)(x^{*-1} + x^{*-2}a + \cdots + a^{*-1})$. The divisibility formula concludes that $\Phi_M(x)$ and $\Phi_{M_j}(x^{M/M_j})$ shares a nontrivial common factor, and the irreducibility of $\Phi_M(x)$ implies that the common factor is $\Phi_M(x)$ itself.

Secondly we check the map is surjective. In particular, x lies in the image of the map. Since $M/M_0, M/M_1, \cdots, M/M_s$ are coprime, integers $r_0, r_1, \cdots, r_s$ can be chosen so that $r_0 M/M_0 + r_1 M/M_1 + \cdots + r_s M/M_s = 1$. In other words, $x_0^{r_0} x_1^{r_1} \cdots x_s^{r_s}$ goes to x. Thus the map, or the restricted one on $\mathcal{R}$, is surjective.

Since both sides have same dimension, here we complete the proof. □

Proof of Lemma 1

Proof. From the isomorphisms above, we can consider a variant of canonical embedding map to a complex tensors:

$$\tau'_{\mathbf{N}_h}(a) = (a({\xi_{M_0}^{g_0}}^{j_0}, \ldots {\xi_{M_s}^{g_s}}^{j_s})) \in \mathbb{C}^{\mathbf{N}_h}$$

where $a \in \mathcal{S}'$, ξ_{M_i} is M_i-th root of unity, $g_0 = 5$, $0 \le j_0 < N_0/2$, g_i are primitive elements in $\mathbb{Z}_{M_i}^*$, $0 \le j_i < N_i$ for $i = 1, \ldots, s$. The map $\tau'_{\mathbf{N}_h}$ can be written as a composition of maps

$$\tau'_{\mathbf{N}_h} = {\tau'}_{N_0/2}^{(0)} \circ {\tau'}_{N_1}^{(1)} \circ \cdots \circ {\tau'}_{N_s}^{(s)} \tag{2}$$

where ${\tau'}^{(i)}$ is given by a tensor of following linear transforms

$$\Sigma'_i = \begin{bmatrix} \xi_{M_i,0}^0 & \xi_{M_i,0}^1 & \cdots & \xi_{M_i,0}^{N_i-1} \\ \xi_{M_i,1}^0 & \xi_{M_i,1}^1 & \cdots & \xi_{M_i,1}^{N_i-1} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{M_i,N_i-1}^0 & \xi_{M_i,N_i-1}^1 & \cdots & \xi_{M_i,N_i-1}^{N_i-1} \end{bmatrix} \tag{3}$$

and I_j the identity matrix of size N_j, where $\xi_{M_i,j} = \exp(\frac{2\pi i \cdot g_i{}^j}{M_i})$.

By using the formula of the linear transforms, we can compare norms; $\|a\|_2^{\mathsf{can}} \le (\prod_i \|\Sigma_i'\|) \, \|a\|_2$, $\|a\|_2 \le \left(\prod_i \|\Sigma_i'\|^{-1}\right) \|a\|_2^{\mathsf{can}}$

where $\|L\|$ for a linear operator L on a complex-valued space is given by the supremum of $\|Lx\|/\|x\|$ along all x. In above, it's square is the sum of maginitude squares of all components in the matrix, or just $\mathrm{Tr}(L^*L)$.

$\Sigma_i'^{-1}$ has components

$$l_{ab} = (-1)^{N_i - a} \frac{e_{N_i - a}(\overline{\xi}_{M_i,b})}{\prod_{c \ne b}(\xi_{M_i,b} - \xi_{M_i,c})} \tag{4}$$

For the p^k-th cyclotomic polynomial

$$\Phi_{p^k}(x) = \Phi_p(x^{p^{k-1}}) = x^{p^{k-1}(p-1)} + x^{p^{k-1}(p-2)} + \cdots + x^{p^{k-1}} + 1$$

, the roots $\xi_1, \cdots, \xi_{p^{k-1}(p-1)}$, and an index $b = 1, 2, \cdots, p^{k-1}(p-1)$, we have

$$\frac{d}{dx}\left((x^{p^{k-1}} - 1)\Phi_{p^{k-1}}(x)\right) = p^{k-1}x^{p^{k-1}-1}\Phi_{p^{k-1}}(x) + (x^{p^{k-1}} - 1)\frac{d}{dx}\Phi_{p^{k-1}}(x)$$

$$\Phi'_{p^{k-1}}(\xi_b) = \frac{p^k \xi_b^{p^k - 1}}{\xi_b^{p^{k-1}} - 1}$$

where $\overline{\xi}_b$ is a vector consisting of all roots but ξ_b of Φ_p and $e_j(\overline{x})$ is an elementary symmetric polynomial of degree j in $p-2$ variables. Note that the denominator is of form 'p-th root of unity -1', not depending on k.

For $N = \phi(p^k) = p^k - p^{k-1}$, $(-1)^{N-a} e_{N-a}(\overline{\xi}_b)$ is the degree a coefficient of

$$\prod_{c \ne b}(x - \xi_c) = \frac{\Phi_{p^k}(x)}{x - \xi_b}$$

, which is in fact $\xi_b^{N-a-[N-a]}(1 + \xi_b^{p^{k-1}} + \cdots + \xi_b^{[N-a]})$ with $[N-a]$ is the largest multiple of p^{k-1} less or eqaul to $N-a$.

In other hands,

$$\Phi'_{p^{k-1}}(\xi_k) = \prod_{l \ne k}(\xi_k - \xi_l)$$

which is the denominator of the formula 4.

Therefore we have

$$\|\Sigma_i'^{-1}\| = \sum_{a,b} |l_{ab}|^2 = \sum_{a,b} \left| \frac{1 - \xi_{M_i,b}^{N_i - a}}{p_i^k \xi_{M_i,b}^{p_i^{k-1}}} \right|^2 = \frac{N_i}{p_i^{2k}} \sum_{a \mod N_i} |1 - \zeta^{N_i - a}|^2$$

where ζ is any primitive p_i-th (NOT p_i^k-th) root of unity . The right-hand side is in fact

$$(\text{if } k_i > 1) \quad \frac{N_i^2}{p_i^{2k+1}} \sum_{i=1}^{p_i - 1} (2 - 2\cos 2\pi i/p_i)$$

$$(\text{if } k_i = 1) \quad \frac{N_i}{p_i^2} \sum_{i=1}^{p_i - 1} (2 - 2\cos 2\pi i/p_i)$$

and since

$$\begin{aligned}
\frac{1}{p}\sum_{i=1}^{p-1}\cos(2\pi i/p) &= \frac{1}{p}\left(\sum_{i=1}^{(p-1)/2}\cos(2\pi i/p) + \sum_{i=(p+1)/2}^{p-1}\cos(2\pi i/p)\right) \\
&\geq \int_{2\pi/p}^{2\pi(p+1)/2p}\cos(x)\ dx + \int_{2\pi(p-1)/2p}^{2\pi(p-1)/p}\cos(x)\ dx \\
&= \int_0^{2\pi}\cos(x)\ dx - 2\int_0^{2\pi/p}\cos(x)\ dx + \int_{2\pi(p-1)/2p}^{2\pi(p+1)/2p}\cos(x)\ dx \\
&\geq -2\times 2\pi/p - 2\pi/p = -6\pi/p
\end{aligned}$$

for any integer p, we conclude that

$$\|\Sigma_i'^{-1}\|^2 \leq \frac{p_i - 1}{p_i} \times (2 + 12\pi/p_i).$$

$\|a(\mathbf{x})\|_2$ is the ℓ_2-norm of a vector whose components consist of the coefficients of $a(\mathbf{x})$. By applying canonical embedding only on x_s, we get a new vector whose components consist of the coefficients of a polynomial $a(x_0, \cdots, x_{s-1}, \xi_s)$ in s variables $x_0, \cdots, x_{s-1}$ and their conjugations. The ℓ_2 norm of the new vector is given by $\Sigma_s^{-1}\cdot$ (coefficient vector of $a(\mathbf{x})$), thus is bounded by $\|\Sigma_s^{-1}\|\|a\|_2$. By induction on s, we have the total bound of $\|a\|_\infty^{\mathsf{can}}{}_2$ as $\prod_{i=0}^{s}\|\Sigma_i'^{-1}\|$. $p_0 = 2$ in our case and it has a special bound $\|\Sigma_0'^{-1}\| = 1$ so that our bound is in fact $\prod_{i=1}^{s}\|\Sigma_i'^{-1}\|$ as desired. □

Proof of Lemma 2.

Proof. We choose $v \leftarrow \mathcal{ZO}(\rho)$, $e_0, e_1 \leftarrow \mathcal{DG}(\sigma)$, then set $\mathsf{ct} \leftarrow v\cdot\mathsf{pk}+(e_0, e_1+m)$. The bound δ_{clean} of encryption noise is computed by the following inequality:

$$\begin{aligned}
\|\langle\mathsf{ct},\mathsf{sk}\rangle - m \pmod{2^L}\|_\infty^{\mathsf{can}} &= \|v\cdot e + e_1 + e_0\cdot s\|_\infty^{\mathsf{can}} \\
&\leq \|v\cdot e\|_\infty^{\mathsf{can}} + \|e_1\|_\infty^{\mathsf{can}} + \|e_0\cdot s\|_\infty^{\mathsf{can}} \\
&\leq 8\sqrt{2}\cdot\sigma N + 6\sigma\sqrt{N} + 16\sigma\sqrt{hN}.
\end{aligned}$$

□

Proof of Lemma 3.

Proof. It is satisfied that $\langle\mathsf{ct},\mathsf{sk}\rangle = m + e \pmod{2^\ell}$ for some polynomial $e \in \mathcal{S}$ such that $\|e\|_\infty^{\mathsf{can}} < \delta$. The output ciphertext $\mathsf{ct}' \leftarrow \lfloor 2^{-p}\cdot\mathsf{ct}\rceil$ satisfies $\langle\mathsf{ct}',\mathsf{sk}\rangle = 2^{-p}\cdot(m+e) + e_{\mathsf{scale}} \pmod{2^{\ell-p}}$ for the rounding error vector $\tau = (\tau_0, \tau_1) = \mathsf{ct}' - 2^{-p}\cdot\mathsf{ct}$ and the error polynomial $e_{\mathsf{scale}} = \langle\tau,\mathsf{sk}\rangle = \tau_0\cdot s + \tau_1$.

We may assume that each coefficient of τ_0 and τ_1 in the rounding error vector is computationally indistinguishable from the random variable in the interval

$2^{-p} \cdot \mathbb{Z}_{2^p}$ with variance $\approx 1/12$. Hence, the magnitude of scale error polynomial is bounded by

$$\|e_{\mathsf{scale}}\|_\infty^{\mathsf{can}} \le \|\tau_0 \cdot s\|_\infty^{\mathsf{can}} + \|\tau_1\|_\infty^{\mathsf{can}} \le 6\sqrt{N/12} + 16\sqrt{hN/12}$$

as desired. $\square$

Proof of Lemma 4.

Proof. Let $\mathsf{ct}_i = (a_i, b_i)$ for $i = 1, 2$. Then $\langle \mathsf{ct}_i, \mathsf{sk}\rangle = m_i + e_i \pmod{2^\ell}$ for some polynomials $e_i \in \mathcal{S}$ such that $\|e_i\|_\infty^{\mathsf{can}} \le \delta_i$. Let $(d_0, d_1, d_2) = (a_1a_2, a_1b_2 + a_2b_1, b_1b_2)$. This vector can be viewed as an encryption of $m_1 \cdot m_2$ with an error $m_1 \cdot e_2 + m_2 \cdot e_1 + e_1 \cdot e_2$ with respect to the secret vector $(s^2, s, 1)$. It follows from Lemma 3 that the ciphertext $\mathsf{ct}_{\mathsf{mult}} \leftarrow (d_1, d_2) + \lfloor 2^{-L} \cdot (d_0 \cdot \mathsf{evk} \pmod{2^{\ell+L}})\rceil$ contains an additional error $e'' = 2^{-L} \cdot d_0 e'$ and a rounding error bounded by δ_{scale}. We may assume that d_0 behaves as a uniform random variable on $\mathcal{R}_\ell$, so $2^L\|e''\|_\infty^{\mathsf{can}}$ is bounded by $16\sqrt{Nq_\ell^2/12}\sqrt{N\sigma^2} = 8N\sigma q_\ell/\sqrt{3} = \delta_{\mathsf{ks}} \cdot 2^\ell$. Therefore, $\mathsf{ct}_{\mathsf{mult}}$ is an encryption of $m_1 \cdot m_2$ with an error and the error is bounded by

$$\|m_1e_2 + m_2e_1 + e_1e_2 + e''\|_\infty^{\mathsf{can}} + \delta_{\mathsf{scale}} \le \\ \mu_1\delta_2 + \mu_2\delta_1 + \delta_1\delta_2 + 2^{-L} \cdot 2^\ell \cdot \delta_{\mathsf{ks}} + \delta_{\mathsf{scale}}$$

as desired. $\square$

Proof of Lemma 5.

Proof. Let prove the lemma for conjugation, proofs of others are the same. The vector $(a', b') = (\kappa_{-\mathbf{1}}(a), \kappa_{-\mathbf{1}}(b)) \pmod{2^\ell}$ can be viewed as an encryption of $\bar{\mathbf{Z}}$ with and error $\kappa_{-\mathbf{1}}(e)$ with respect to the secret vector $(\kappa_{-\mathbf{1}}(s), 1)$. Using proof of Lemma 4 we can get that $\mathsf{ct}_{\mathsf{cj}}$ is an encryption of $\bar{\mathcal{Z}}$ with an error bounded by

$$\|\kappa_{-1,1}(e) + e''\|_\infty^{\mathsf{can}} + \delta_{\mathsf{scale}} \le \delta + 2^{-L} \cdot 2^\ell \cdot \delta_{\mathsf{ks}} + \delta_{\mathsf{scale}}$$

as desired. $\square$

Proof of Lemma 6.

Proof. From Lemma 5 and the following remark about the relative error we can see that bound of message increase only after summations in line 10 of Algorithm 5, so the bound M of the output is equal to $n \cdot 2^p$. Note also that these summations do not increase the bound of the relative error. The relative error increases by β^* after rotation and increases by β^* after multiplication. So the relative error of each summand in line 10 is bounded by $\beta_{\mathbf{A}} + \beta_{\mathbf{B}} + (1 + \log n)\beta^*$. $\square$

Proof of Lemma 7.

Proof. The relative error increases by β^* after rotation. So the relative error of each summand $\mathsf{ct}_{\mathbf{A}_k}$ is bounded by $\beta_{\mathbf{A}} + \beta^*$. The relative error we can see that bound of message and bound of relative error does not increase during summations of $\mathsf{ct}_{\mathbf{A}_k}$. □

Proof of Lemma 8.

Proof. From Lemma 6 the message of $\mathsf{ct}_{\mathbf{A}_j}$ is bounded by $\epsilon^{2^j} 2^p/n$ which implies that the message of $\mathsf{ct}_{\mathbf{V}_r}$ is bounded by

$$2^{p-t} \prod_{j=0}^{r-1} (1 + \epsilon^{2^j}/n) < \frac{2^{p-t}}{(1-\epsilon)^{1/n}} < n^{1/n} 2^{p-t}$$

The relative error β_j of $\mathsf{ct}_{\mathbf{A}_j}$ is bounded by $\beta_j \leq 2^j(\beta + (1 + \log n)\beta^*)$, which implies that the relative error β'_j of $\mathsf{ct}_{\mathbf{A}_j} + i$ is bounded by

$$\beta'_j \leq \beta_j / \Big(1 + \frac{n}{\epsilon^{2^j}}\Big)$$

Using induction on j, we can show that a relative error β''_j of $\mathsf{ct}_{\mathbf{V}_j}$ is bounded by

$$\begin{aligned}
&\beta''_j \leq \Big(\sum_{k=0}^{j-1} \frac{2^k \epsilon^{2^k}}{n + \epsilon^{2^k}}\Big) \cdot (\beta + (1 + \log n) \cdot \beta^*) + (j-1) \cdot (1 + \log n) \cdot (\beta^*) \leq \\
&\frac{1}{n} \sum_{k=0}^{j-1} (2^k \epsilon^{2^k}) \cdot (\beta + (1 + \log n) \cdot \beta^*) + (j-1) \cdot (1 + \log n) \cdot \beta^* \leq \\
&\frac{2}{n(1-\epsilon)} \cdot (\beta + (1 + \log n) \cdot \beta^*) + (j-1) \cdot (1 + \log n) \cdot \beta^* \leq \\
&2\beta + (j+1) \cdot (1 + \log n) \cdot \beta^*
\end{aligned}$$

□