Homomorphic Encryption for Approximate Matrix Arithmetic

Jung Hee Cheon\textsuperscript{1}, Andrey Kim\textsuperscript{1}

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

Abstract. Homomorphic Encryption for Arithmetic of Approximate Numbers (HEAAN) with its vector packing technique proved its potential in cryptographic applications. In this paper, we propose MHEAAN - a generalization of the HEAAN scheme to a multivariate case. 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 arithmetic, such as floating point operations. In addition, with a multivariate structure of the plaintext space, we suggest a general method of packing multidimensional structures as matrices and tensors in a single ciphertext. We provide a concrete two-dimensional construction and show the efficiency of our scheme on several matrix operations, such as matrix transposition, matrix multiplication, and inverse.

Keywords. Homomorphic encryption, approximate arithmetic, matrix multiplication, matrix inverse, transposition

1 Introduction

Homomorphic encryption (HE) is a scheme that allows to perform certain arithmetic operations in encrypted state. Following Gentry’s blueprint [19] a numerous HE schemes have been proposed (e.g. [14, 5, 6, 3, 4, 20, 27, 2, 21, 12, 11, 16, 15, 9]). HE has plenty applications in evaluation of various algorithms on encrypted financial, medical, or genome data [29, 26, 10, 37, 25, 24].

Packing is a common technique in cryptographic schemes that allows to encrypt several messages in a single ciphertext, providing more potential to SIMD operations. Most of cryptographic schemes provide a vector packing, along with rotation operations among slots. Some HE, such as HElib [22] also provide packing messages withing a hypercube structure, however this packing technique highly depend on the structure of underlying primes factors of a modulus and is not very intuitive.

Recently Cheon et. al. [9] presented a method to construct a HE scheme for arithmetic of approximate numbers (named HEAAN as follows). The idea of the construction is to treat an encryption noise as part of an error occurring during approximate computations. That is, an encryption $ct$ of a plaintext $m \in \mathcal{R}$ by a secret key $sk$ for an underline ciphertext modulus $q$ will have a decryption structure of the form $(ct, sk) = m + e \mod q$ for some small error $e$. HEAAN
showed its potential by providing the winning solution of Track 3 (Homomorphic encryption based logistic regression model learning) at iDASH privacy and security competition 2017 [24]. 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 additional level during the computations.

Despite a diversity of HE schemes that achieve a variety of circuits evaluation, practical matrix operations such as matrix multiplication is still a problem in HE. 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 becoming crucial for Homomorphic Encryption. Some works as Duong et al. [17] suggested a method for a single matrix multiplication using special packing methods but its packing structure seems to have problems with expanding to more complex computations. Another work of Hiromasa et al. [23] constructed a matrix version of GSW scheme, however it only consider binary matrix case and requires a lot of matrix multiplications, and seems not to be practical.

In this paper we suggest a generalization of HEAAN scheme with a new tensor packing method, along with natural rotations in various dimensions and transposition operations. Our construction (named MHEAAN as follows) is based on a Multivariate RLWE (m-RLWE) problem as an underline hard problem. Multivariate RLWE (m-RLWE) was introduced by Pedrouzo-Ulloa et al. [30, 31] as a multidimensional variant of RLWE problem. The hardness of m-RLWE is related to hardness problems over the tensor product of ideal lattices, and is supposed to be more appropriate for cryptographic applications dealing with multidimensional structures [32] including matrices.

Thus MHEAAN scheme enjoys all the benefits of HEAAN scheme such as rescaling procedure, which enables us to preserve the precision of the message after approximate computation and reduce the size of ciphertext significantly so the scheme can be a reasonable solution for approximate computation over the complex values. Moreover with a multivariate structure instead of packing a vector, we provide a general technique for packing any dimensional tensor in a single ciphertext. In particular we constructed a practical HE which supports matrix operations as well as standard HE operations. For example it takes only 67 ms to evaluate a transposition of a $64 \times 64$ complex matrix. For homomorphic multiplication of two $64 \times 64$ complex matrices it takes about 29 seconds, and about 4 minutes to evaluate 16-th power of a $64 \times 64$ complex matrix. Matrix inverse is a more complicated problem even in unencrypted case [36, 35]. We provided a method to evaluate inverse of a matrix for a special case. It takes about 7 minutes for a $64 \times 64$ complex matrix.


2 Preliminaries

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. We denote by \(\langle \mathbf{a}, \mathbf{b} \rangle\) the usual dot product of two vectors. By bold capital letters we denote matrices, e.g. \(\mathbf{A}\). For a real number \(r\), \(|r|\) 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 interval and use \([r]_q\) to denote the reduction of an integer \(r\) modulo \(q\) into that interval. We use \(a \leftarrow \chi\) to denote the sampling according to a distribution \(\chi\). For a finite set \(D\), \(a \leftarrow D\) denotes the sampling from the uniform distribution over \(D\). For matrices \(\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n \times m}\) denote by \(\mathbf{A} \odot \mathbf{B}\) component wise product, and denote by \(\text{rt}(\mathbf{A}, (r, c))\) a matrix obtained from \(\mathbf{A}\) by cyclic rotating by \(r\) rows and \(c\) columns.

\[
\mathbf{A} = \begin{bmatrix}
a_{0,0} & a_{0,1} & \cdots & a_{0,m-1} \\
a_{1,0} & a_{1,1} & \cdots & a_{1,m-1} \\
& & \cdots & \\
a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,m-1}
\end{bmatrix}
\]

\[
\text{rt}(\mathbf{A}, (r, c)) = \begin{bmatrix}
a_{r,c} & a_{r,c+1} & \cdots & a_{r,c+m-1} \\
a_{r+1,c} & a_{r+1,c+1} & \cdots & a_{r+1,c+m-1} \\
& & \cdots & \\
a_{r+n-1,c} & a_{r+n-1,c+1} & \cdots & a_{r+n-1,c+m-1}
\end{bmatrix}
\]

where indexes are taken modulus \(n\) and \(m\) respectively. 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 a power of two \(N\), let \(M = 2N\) and \(\Phi_M(X) = X^N + 1\) be \(M\)-th cyclotomic polynomial. Let \(\mathcal{S} = \mathbb{Z}[X]/(X^N + 1)\). An element in \(\mathcal{S}\) can be represented as a polynomial \(a(X) = \sum_{j=0}^{N-1} a_jX^j\) of degree less than \(N\) and identified with its coefficient vector \((a_0, \ldots, a_{N-1})\) \(\in \mathbb{R}^N\). We define the relevant norms on the coefficient vector of \(a\) such as \(\|a\|_\infty\) and \(\|a\|_1\).

Let \(\mathbb{Z}_M^* = \{x \in \mathbb{Z}_M : \gcd(x, M) = 1\}\) be the multiplicative group of units in \(\mathbb{Z}_M\). The canonical embedding \(\sigma_M\) of \(a \in \mathbb{Q}[X]/(\Phi_M(X))\) into \(\mathbb{C}^N\) is the vector of evaluation values of \(a\) at the roots of \(\Phi_M(X)\). We naturally extend it to the set of real polynomials \(\mathcal{S}\), \(\sigma_M : \mathcal{S} \rightarrow \mathbb{C}^N\), so \(\sigma_M(a)\) will be defined as \((a(\xi_M^j))_{j \in \mathbb{Z}_M} \in \mathbb{C}^N\) for any \(a \in \mathcal{S}\) where \(\xi_M = \exp(-2\pi i/M)\) is a primitive \(M\)-th roots of unity. The \(\ell_\infty\)-norm of \(\sigma_M(a)\) is called the canonical embedding norm of \(a\), denoted by \(\|a\|_{\infty} = \|\sigma_M(a)\|_{\infty}\). The canonical embedding norm \(\|a\|_{\infty}^{2^n}\) satisfies the following properties:
For all $a, b \in \mathbb{R}$, we have $\|a \cdot b\|_{\infty} \leq \|a\|_{\infty} \cdot \|b\|_{\infty}$

- For all $a \in \mathbb{R}$, we have $\|a\|_{\infty} \leq \|a\|_{1}$.
- For all $a \in \mathbb{R}$, we have $\|a\|_{\infty} \leq \|a\|_{c_{\infty}}$.


2.3 Learning With Errors and its Variants

Learning With Errors (LWE) problem introduced by Regev [34], proved itself as a good underlying problem for many lattice-based cryptography schemes.

**Definition 1 (LWE Distribution).** Let $N > 1$, $q \geq 2$. For $s \in \mathbb{Z}_q$ (the “secret”) and an error distribution $\chi$ over $\mathbb{Z}_q$, a sample from the LWE distribution $A_{s, \chi}$ over $\mathbb{Z}_q^N \times \mathbb{Z}_q$ is generated by choosing $a \leftarrow \mathbb{Z}_q^N$ uniformly at random, choosing $e \leftarrow \chi$, and outputting $(a, \langle a, s \rangle + e \mod \mathbb{Z}_q)$.

**Definition 2 (LWE Search).** The search version of the LWE problem is defined as follows: given access to arbitrary number of independent samples from LWE distribution $A_{s, \chi}$ for some arbitrary $s \in \mathbb{Z}_q$ find $s$ (with high probability).

Even though LWE has promising hardness assumption for cryptographic primitives [34, 33], it is not efficient enough for practical applications. The Ring LWE (RLWE) introduced by Lyubashevsky et al. [28] is an algebraic variant of LWE which provides more attractive features for cryptographic applications. We provide definition of RLWE for a power-of-two degree case.

**Definition 3 (RLWE Distribution).** Let $N > 1$ be a power-of-two, $q \geq 2$, $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$, $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. For $s \in \mathcal{R}$ (the “secret”) and an error distribution $\chi$ over $\mathcal{R}_q$, a sample from the RLWE distribution $A_{s, \chi}$ over $\mathcal{R}_q \times \mathcal{R}_q$ is generated by choosing $a \leftarrow \mathcal{R}_q$ uniformly at random, choosing $e \leftarrow \chi$, and outputting $(a, a \cdot s + e \mod \mathcal{R}_q)$.

**Definition 4 (RLWE Search).** The search version of the RLWE problem is defined as follows: given access to arbitrary number of independent samples from RLWE distribution $A_{s, \chi}$ for some arbitrary $s \in \mathcal{R}$ find $s$ (with high probability).

Multivariate RLWE (m-RLWE) was introduced by Pedrouzo-Ulloa et al. [30, 31] as a multidimensional variant of RLWE problem. m-RLWE hardness is related to hardness problems over the tensor product of ideal lattices, and is supposed to be more appropriate for cryptographic applications dealing with multidimensional structures [32].

**Definition 5 (m-RLWE Distribution).** Let $N_i > 1$ be a powers-of-two for $i = 1, \ldots, l$, $q \geq 2$, $\mathcal{R} = \mathbb{Z}[X_1, \ldots, X_l]/(X_1^N + 1, \ldots, X_l^N + 1)$, $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. For $s \in \mathcal{R}$ (the “secret”) and an error distribution $\chi$ over $\mathcal{R}_q$, a sample from the m-RLWE distribution $A_{s, \chi}$ over $\mathcal{R}_q \times \mathcal{R}_q$ is generated by choosing $a \leftarrow \mathcal{R}_q$ uniformly at random, choosing $e \leftarrow \chi$, and outputting $(a, a \cdot s + e \mod \mathcal{R}_q)$.

**Definition 6 (m-RLWE Search).** The search version of the m-RLWE problem is defined as follows: given access to arbitrary number of independent samples from m-RLWE distribution $A_{s, \chi}$ for some arbitrary $s \in \mathcal{R}$ find $s$ (with high probability).
2.4 HEAAN Scheme

The following is the instantiation of the RLWE-based HEAAN scheme [9, 8]. For a power-of-two \( N \), denote \( M = 2N \), \( \Phi_M(X) = (X^N + 1) \), \( \mathcal{R} = \mathbb{Z}[X]/\Phi_M(X) \).

For a positive integer \( \ell \), 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 \( 2^\ell \). The integer 5 has the order of \( N/2 \) modulo \( M \) and with \((-1)\) spans \( \mathbb{Z}_M^* \). Hence \( \xi_j, \bar{\xi}_j : 0 \leq j < N/2 \) forms the set of primitive \( M \)-th roots of unity for \( \xi_j := \xi_j^5 \). Define an encoding map \( \phi_{N/2} \) as inverse of a variant of the complex canonical embedding map defined as \( \phi_{N/2} : m(x) \rightarrow z = (z_0, \ldots, z_{N/2-1}) \) such that \( z_j = m(\xi_j) \).

For a power-of-two \( n \leq N/2 \) for \( X' = X^{N/(2n)} \) consider \( \mathcal{R}' = \mathbb{Z}[X']/(X'^{2n} + 1) \subset \mathcal{R} \) and define in a similar way \( \phi_n^{-1} : m(x') = m(x^{N/(2n)}) \rightarrow z = (z_0, \ldots, z_{n-1}) \) such that \( z_j = m(\xi'_j) \), where \( \xi'_j = \xi^j_{N/(2n)} \).

- **HEAAN.KeyGen(1^\lambda).**
  - For an integer \( L \) that corresponds to the largest ciphertext modulus level,
    given the security parameter \( \lambda \), output the ring dimension \( N \) which is a power of two.
  - Set the small distributions \( \chi_{\text{key}}, \chi_{\text{err}}, \chi_{\text{enc}} \) over \( \mathcal{R} \) for secret, error, and encryption, respectively.
  - Sample a secret \( s \leftarrow \chi_{\text{key}} \), a random \( a \leftarrow \mathcal{R}_L \) and an error \( e \leftarrow \chi_{\text{err}} \).
  - Set the secret key as \( \text{sk} \leftarrow (s, 1) \) and the public key as \( \text{pk} \leftarrow (a, b) \in \mathcal{R}_L^2 \)
    where \( b \leftarrow -as + e \pmod{2^L} \).

- **HEAAN.KSGen\(_{\text{s}}(s^2)\).**
  For \( s^2 \in \mathcal{R} \), sample a random \( a' \leftarrow \mathcal{R}_{2^L} \) and an error \( e' \leftarrow \chi_{\text{err}} \).
  Output the switching key as \( \text{swk} \leftarrow (a', b') \in \mathcal{R}_{2^L}^2 \) where \( b' \leftarrow -a's + e' + 2^Ls' \pmod{2^{2L}} \).
  - Set the evaluation key as \( \text{evk} \leftarrow \text{HEAAN.KSGen}_{a}(s^2) \).

- **HEAAN.Encode(z, p).** For a vector \( z \in \mathbb{C}^n \), with \( p \in (\text{of two} n \leq N/2 \) and an integer \( p < L \) corresponding to precision bits, output the polynomial \( m \leftarrow \phi_{n}^{-1}(m/p) \in \mathbb{C}^n \).

- **HEAAN.Decode(m, p).** For a plaintext \( m \in \mathcal{R} \), the encoding of a vector consisting of \( p \leq N/2 \) complex messages and precision bits \( p \), output the vector \( z \leftarrow \phi_{n}^{-1}(m/p) \in \mathbb{C}^n \).

- **HEAAN.Enc\(_{\text{pk}}(m)\).** For \( m \in \mathcal{R} \), sample \( v \leftarrow \chi_{\text{enc}} \) and \( e_0, e_1 \leftarrow \chi_{\text{err}} \).
  Output \( v \cdot \text{pk} + (e_0, e_1 + m) \pmod{2^L} \).

- **HEAAN.Dec\(_{\text{sk}}(ct)\).** For \( ct = (c_0, c_1) \in \mathcal{R}_L^2 \), output \( c_0 \cdot s + c_1 \pmod{2^L} \).

- **HEAAN.Add(\text{ct}_1, \text{ct}_2).** For \( \text{ct}_1, \text{ct}_2 \in \mathcal{R}_L^2 \), output \( \text{ct}_{\text{add}} \leftarrow \text{ct}_1 + \text{ct}_2 \pmod{2^L} \).

- **HEAAN.CMult\(_{\text{evk}}(\text{ct}, c, p).** For \( \text{ct} \in \mathcal{R}_L^2 \) and \( c \in \mathbb{C}^n \), compute \( c \leftarrow \text{HEAAN.Encode}(c; p) \)
  and output \( \text{ct}' \leftarrow c \cdot \text{ct} \pmod{2^L} \).

- **HEAAN.PolyMult\(_{\text{evk}}(\text{ct}, g, p).** For \( \text{ct} \in \mathcal{R}_L^2 \) and \( g \in \mathcal{R}_L \), output \( \text{ct}' \leftarrow g \cdot \text{ct} \pmod{2^L} \).

- **HEAAN.Mult\(_{\text{evk}}(\text{ct}_1, \text{ct}_2).** For \( \text{ct}_1 = (a_1, b_1), \text{ct}_2 = (a_2, b_2) \in \mathcal{R}_L^2 \), let \( d_0, d_1, d_2 \) = \( (a_1 \cdot a_2, a_1 \cdot b_2 + a_2 \cdot b_1, b_1 \cdot b_2) \pmod{2^L} \).
  Output \( \text{ct}_{\text{mult}} \leftarrow (d_1, d_2) + [2^{-L} \cdot d_0 \cdot \text{evk}] \pmod{2^L} \).
...for ciphertext \(ct \in \mathcal{R}_F^2\) and an integer \(p\), output \(ct' \leftarrow \lfloor 2^{e-p} \cdot ct \rfloor \mod 2^{2\ell - p}\).

- **HEAAN.ModDown(\(ct, p\)).** For a ciphertext \(ct \in \mathcal{R}_F^2\) and an integer \(p\), output \(ct' \leftarrow ct \mod 2^{2\ell - p}\).

For an integer \(k\) co-prime with \(M\), let \(\kappa_k : m(X) \rightarrow m(X^k) \pmod {\Phi_M(X)}\).

This transformation can be used to provide more functionalities on plaintext slots.

- **HEAAN.Conjugate\(_{cjk}\)(\(ct\)).** Set the conjugation key as \(c_{jk} \leftarrow \text{HEAAN.KSGen}_k(\kappa_{-1}(s))\).

For \(ct = (a, b) \in \mathcal{R}_F^2\) encrypting vector \(z\), let \((a', b') = (\kappa_{-1}(a), \kappa_{-1}(b)) \pmod {2^\ell}\). Output \(ct_{cjk} \leftarrow (0, b') + [2^{-L} \cdot a' \cdot c_{jk}] \pmod {2^\ell}\). \(ct_{cjk}\) is a ciphertext encrypting \(\bar{z}\) - the conjugated plaintext vector of \(ct\).

- **HEAAN.Rotate\(_{rk}\)(\(ct; r\)).** Set the rotation key as \(rt_k \leftarrow \text{HEAAN.KSGen}_k(\kappa_{5r}(s))\).

For \(ct = (a, b) \in \mathcal{R}_F^2\) encrypting vector \(z\), let \((a', b') = (\kappa_{5r}(a), \kappa_{5r}(b)) \pmod {2^\ell}\). Output \(ct_{rt} \leftarrow (0, b') + [2^{-L} \cdot a' \cdot rt_k] \pmod {2^\ell}\). \(ct_{rt}\) is a ciphertext encrypting \(rt(z, r) = (z_r, \ldots, z_{n-1}, z_0, \ldots, z_{r-1})\) - rotated by \(r\) positions plaintext vector of \(ct\).

Refer [9] for the technical details and noise analysis.

### 3 Multivariate HEAAN Scheme

#### 3.1 Structure of MHEAAN

Encryption process in MHEAAN scheme can be shown in the following outline: for a powers-of-two \(N_i, n_i, \text{ with } n_i \leq N_i/2, i = 1, \ldots, l\) encode a \(l\)-dimension tensor \(Z \in \mathbb{C}^{n_1 \times \cdots \times n_l}\) to a \(l\)-variable polynomial \(m(X_1, \ldots, X_l)\), using map \(\phi_{n_1, \ldots, n_l}\) and mask the result with \(m\)-RLWE instance \((a(X_1, \ldots, X_l), b(X_1, \ldots, X_l))\) in the corresponding ring \(\mathcal{R}/q\mathcal{R}\).

We can treat HEAAN scheme as a special case of MHEAAN with \(l = 1\):
In the rest of the paper we will use following notations. For a power-of-two integers $N_x, N_y$ denote $M_x = 2N_x, M_y = 2N_y, \Phi_{M_x}(X) = X^{N_x} + 1, \Phi_{M_y}(Y) = Y^{N_y} + 1, \mathcal{R} = \mathbb{Z}[X,Y]/(\Phi_{M_x}(X), \Phi_{M_y}(Y)), S = \mathbb{R}[X,Y]/(\Phi_{M_x}(X), \Phi_{M_y}(Y))$.

We can generalize $\sigma_{M_x}$ defined in Section 2 to $\sigma_{M_x,M_y} : \mathcal{S} \rightarrow \mathbb{C}^{N_x,N_y}$, so for $a \in \mathcal{S}, \sigma_{M_x,M_y}(a) = (a(\xi_x^{j_x}, \xi_y^{j_y}))_{j_x \in \mathbb{Z}_{M_x}, j_y \in \mathbb{Z}_{M_y}} \in \mathbb{C}^{N_x \times N_y}$. Two dimensional canonical embedding norm is defined as $\|a\|_{mcan}^\infty = \|\sigma_{M_x,M_y}(a)\|_{\infty}$. $\| \cdot \|_{\infty}$ satisfies the following properties:

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

For a power-of-two integers $n_x \leq N_x/2$ and $n_y \leq N_y/2$, let $X' = X^{N_x/(2n_x)}, Y' = Y^{N_y/(2n_y)}$ and $\mathcal{R}' = \mathbb{Z}[X',Y']/((X'^{2n_x}+1,Y'^{2n_y}+1)$. We can also generalize encoding map $\phi_n$ to a two-dimensional mapping $\phi_{n_x,n_y} : \mathbb{C}^{n_x \times n_y} \rightarrow \mathcal{R}$, defined as $\phi_{n_x,n_y}^{-1}(X', Y') = m(x^{N_x/(2n_x)}, y^{N_y/(2n_y)}) \rightarrow \mathcal{Z}$

where $\mathcal{Z} \in \mathbb{C}^{n_x \times n_y}$ with $\mathcal{Z}_{ij} = m(\xi_x^{j_x}, \xi_y^{j_y}, \xi_x^{j_x}, \xi_y^{j_y}) = \xi_x^{N_x/(2n_x)}, \xi_y^{N_y/(2n_y)}$.

### 3.3 Concrete Construction

We present a construction of 2-RRLWE-based MHEAN scheme. For a positive integer $\ell$ denote $\mathcal{R}_\ell = \mathcal{R}/2^\ell \mathcal{R} = \mathbb{Z}_{2^\ell}[X,Y]/(\Phi_{M_x}(X), \Phi_{M_y}(Y))$ the residue ring of $\mathcal{R}$ modulo $2^\ell$. For a real $\sigma > 0$, $\mathcal{D}_\mathcal{G}(\sigma^2)$ samples a polynomial in $\mathcal{R}$ by drawing its coefficient independently from the discrete Gaussian distribution of variance $\sigma^2$. For an positive integer $h$, $\mathcal{H}_W T(h)$ is the set of signed binary matrices in $\{0, \pm 1\}^{N_x \times N_y}$ whose Hamming weight is exactly $h$. For a real $0 \leq \rho \leq 1$, the distribution $\mathcal{Z}_\rho$ draws each entry in the matrix from $\{0, \pm 1\}^{N_x \times N_y}$, with probability $\rho/2$ for each of $-1$ and $+1$, and probability being zero $1 - \rho$. 

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

- **MHEAAN.KeyGen(1)**.
  - For an integer $L$ that corresponds to the largest ciphertext modulus level, given the security parameter $\lambda$, distribution parameters $(\rho, \sigma, h)$, output the power-of-two ring dimension $N$ and choose $N_x, N_y$ such that $N_x \cdot N_y = N$
  - 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 $sk \leftarrow (s, 1)$ and the public key as $pk \leftarrow (a, b) \in \mathcal{R}_L^2$
    where $b \leftarrow -a \cdot s + e \pmod{2^L}$.

- **MHEAAN.KGen_{sk}(s')**.
  - For $s' \in \mathcal{R}$, sample a random $a' \leftarrow \mathcal{R}_{2^L}$ and an error $e' \leftarrow \chi_{err}$.
  - Output the switching key as $swk \leftarrow (a', b') \in \mathcal{R}_L^2$
    where $b' \leftarrow -a' \cdot s + e' + 2^L s' \pmod{2^L}$.
    - Set the evaluation key as $evk \leftarrow \text{MHEAAN.KGen}_{sk}(s)$.

- **MHEAAN.Encode(Z, p)**.
  - For a matrix $Z \in \mathbb{C}^{n_x \times n_y}$, an integer $p < L - 1$ corresponding to precision bits, output the two-degree polynomial $m \leftarrow \phi_{n_x,n_y}(2^p \cdot Z) \in \mathcal{R}$.

- **MHEAAN.Decode(m, p)**.
  - For a plaintext $m \in \mathcal{R}$, the encoding of a matrix of complex messages $Z \in \mathbb{C}^{n_x \times n_y}$, precision bits $p$, output the matrix $Z' \leftarrow \phi_{n_x,n_y}^{-1}(m/2^p) \in \mathbb{C}^{n_x \times n_y}$.

- **MHEAAN.Enc_{pk}(m)**.
  - For $m \in \mathcal{R}$, sample $v \leftarrow \chi_{enc}$ and $e_0, e_1 \leftarrow \chi_{err}$.
  - Output $ct = v \cdot pk + (e_0, e_1 + m) \pmod{2^L}$.

- **MHEAAN.Dec_{sk}(ct)**.
  - For $ct = (c_0, c_1) \in \mathcal{R}_L^2$, output $c_0 \cdot s + c_1 \pmod{2^L}$.

- **MHEAAN.Add(ct_1, ct_2)**.
  - For $ct_1, ct_2 \in \mathcal{R}_L^2$ - encryption of matrices $Z_1, Z_2 \in \mathbb{C}^{n_x \times n_y}$ output $ct_{add} \leftarrow ct_1 + ct_2 \pmod{2^L}$. $ct_{add}$ is a ciphertext encrypting matrix $Z_1 + Z_2$.

- **MHEAAN.CMult_{sk}(ct, C, p)**.
  - For $ct \in \mathcal{R}_L^2$ - encryption of $Z \in \mathbb{C}^{n_x \times n_y}$, and a constant matrix $C \in \mathbb{C}^{n_x \times n_y}$, compute $c \leftarrow \text{MHEAAN.Encode}(C, p)$ the encoding of $C$ and output $ct_{c\cdot C} \leftarrow c \cdot ct \pmod{2^L}$. $ct_{c\cdot C}$ is a ciphertext encrypting matrix $Z \circ C$.

- **MHEAAN.PolyMult_{sk}(ct, g, p)**.
  - For $ct \in \mathcal{R}_L^2$ - encryption of $Z \in \mathbb{C}^{n_x \times n_y}$, and a constant $g \in \mathcal{R}_L$ output $ct_{c\cdot ct} \leftarrow c \cdot ct \pmod{2^L}$. $ct_{c\cdot ct}$ is a ciphertext encrypting matrix $Z \circ C$, where $C$ is decoding of $g$.

Multiplication by polynomial is similar to constant multiplication, however in the next section we will show why it is important to define it separately.
Similar to HEAAN in MHEAAN scheme for an integers $k_x, k_y$ co-prime with $M_x, M_y$ respectively, let

$$\kappa_{k_x,k_y} : m(X,Y) \rightarrow m(X^{k_x},Y^{k_y}) \pmod{\Phi_{M_x}(X),\Phi_{M_y}(Y)}$$

This transformation can be used to provide conjugation, row and column rotations on plaintext matrix.

- **MHEAAN.Conjugate_{cjk}(ct)**. Set the conjugation key as

  $$cjk \leftarrow \text{MHEAAN.KGen}_{sk}(\kappa_{-1,1}(s))$$

  For $ct = (a, b) \in \mathbb{R}_m^2$ encrypting matrix $Z$, let $(a', b') = (\kappa_{-1,1}(a), \kappa_{-1,1}(b)) \pmod{2^\ell}$. Output $ct_{cjk} \leftarrow (0, b') + [2^{-L} \cdot a' \cdot \cdot \cdot cjk] \pmod{2^\ell}$. $ct_{cjk}$ is a ciphertext encrypting $Z$ - the conjugated plaintext matrix of $ct$.

- **MHEAAN.Rotate_{rtk}(ct; (r_x, r_y))**. Set the rotation key as

  $$rtk \leftarrow \text{MHEAAN.KGen}_{sk}(\kappa_{5^{r_x},5^{r_y}}(s))$$

  For $ct = (a, b) \in \mathbb{R}_m^2$ encrypting matrix $Z$, let $(a', b') = (\kappa_{5^{r_x},5^{r_y}}(a), \kappa_{5^{r_x},5^{r_y}}(b)) \pmod{2^\ell}$. Output $ct_{rtk} \leftarrow (0, b') + [2^{-L} \cdot a' \cdot rtk] \pmod{2^\ell}$. $ct_{rtk}$ is a ciphertext encrypting $rt(Z, (r_x, r_y))$ - cyclic rotated plaintext matrix by $r$ rows and $c$ columns.

In case of $N_x = N_y$ let $\tau : m(X,Y) \rightarrow m(Y,X)$. This transformation can be used to provide transposition functionality on plaintext matrix.

- **MHEAAN.Transpose_{trk}(ct)**. Set the transposition key as

  $$trk \leftarrow \text{MHEAAN.KGen}_{sk}(\tau(s))$$

  For $ct = (a, b) \in \mathbb{R}_m^2$ encrypting matrix $Z$, let $(a', b') = (\tau(a), \tau(b)) \pmod{2^\ell}$. Output $ct_{trk} \leftarrow (0, b') + [2^{-L} \cdot a' \cdot trk] \pmod{2^\ell}$. $ct_{trk}$ is a ciphertext encrypting $Z^T$ - transposed plaintext matrix of $ct$.

Throughout this paper, we use real polynomials as plaintexts for convenience of analysis. A ciphertext $ct \in \mathbb{R}_m^2$ will be called a valid encryption of $m \in S$ with the encryption noise bounded by $B$, and plaintext bounded by $M$, if $(ct,sk) = m + e \pmod{2^\ell}$ for some polynomial $e \in S$ with $\|e\|_{\infty} < B$ and $\|m\|_{\infty} < M$. We will use a corresponding tuple $(ct, B, M, \ell)$ 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 1 (Encoding & Encryption).** For $m \leftarrow \text{MHEAAN.Encode}(Z,p)$ and $ct \leftarrow \text{MHEAAN.Enc}_{sk}(m)$ the encryption noise is bounded by $B_{\text{clean}} = 8\sqrt{2} \cdot \sigma N + 6\sigma \sqrt{N} + 16\sigma \sqrt{N}$.

**Lemma 2 (Rescaling).** Let $(ct, B, M, \ell)$ be a valid encryption of $m$ and $ct' \leftarrow \text{MHEAAN.ReScale}(ct, p)$. Then $(ct', B/2^p + B_{\text{scale}}, M/2^p, \ell - p)$ is a valid encryption of $m/2^p$ where $B_{\text{scale}} = 6\sqrt{N/12} + 16\sqrt{hN/12}$.
Remark We can slightly change the public key generation and encryption process, to obtain a ciphertext with initial noise reduced from \(B_{\text{clean}}\) to almost \(B_{\text{scale}}\). For this we generate public key in \(R_{2L}^2\) instead of \(R_{2}^2\) and in encryption process we encode the plaintext \(m\) with \(p + L\) precision bits, instead of \(p\) bits with the following rescaling the encryption \(ct\) of \(m\) by \(L\) bits. With a slightly slower encryption process we end up with a valid encryption in \(R_{L}^2\), with the initial noise bounded by \(B_{\text{clean}} / 2^L + B_{\text{scale}} \approx B_{\text{scale}}\).

Lemma 3 (Addition & Multiplication). Let \((ct_i, B_i, M_i, \ell)\) be encryptions of \(m_i \in S\) and let \(ct_{\text{add}} \leftarrow \text{MHEAAN.Add}(ct_1, ct_2)\) and \(ct_{\text{mult}} \leftarrow \text{MHEAAN.Mult}_{\text{evk}}(ct_1, ct_2)\). Then \((ct_{\text{add}}, B_1 + B_2, M_1 + M_2, \ell)\) and \((ct_{\text{mult}}, M_1 \cdot B_2 + M_2 \cdot B_1 + B_1 \cdot B_2 + B_{\text{mult}}, M_1 \cdot M_2, \ell)\) are valid encryptions of \(m_1 + m_2\) and \(m_1 \cdot m_2\), respectively, where \(B_{\text{ks}} = 8\sigma N / \sqrt{3}\) and \(B_{\text{mult}} = 2^{\ell - L} \cdot B_{\text{ks}} + B_{\text{scale}}\).

Lemma 4 (Conjugation, Rotation & Transposition). Let \((ct, B, M, \ell)\) be encryption of \(m \in S\) that encodes matrix \(Z\), \(r_x, r_y \in \mathbb{Z}\), and let ciphertexts \(ct_{\text{rt}} = \text{MHEAAN.Rotate}_{\text{rk}}(ct, (r_x, r_y))\), \(ct_{\text{cj}} = \text{MHEAAN.Conjugate}_{\text{cj}}(ct)\) and \(ct_{\text{tr}} = \text{MHEAAN.Transpose}_{\text{rk}}(ct)\). Then \((ct_{\text{rt}}, B + B_s, M, \ell), (ct_{\text{cj}}, B + B_s, M, \ell), (ct_{\text{tr}}, B + B_s, M, \ell)\) are valid encryptions of plaintexts that encode matrices \(rt(Z, (r_x, r_y))\), \(Z\) and \(Z^T\) respectively where \(B_{\text{ks}} = 8\sigma N / \sqrt{3}\) and \(B_s = 2^{\ell - L} \cdot B_{\text{ks}} + B_{\text{scale}}\).

Relative Error As was discussed in [9] the decryption result 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 \(\beta = B / M\). For addition of two ciphertexts with relative errors \(\beta_i = B_i / M_i\) the output ciphertext has a relative error bounded by \(\max_i(\beta_i)\). For multiplication of two ciphertexts with the following rescaling by \(p\) bits the output ciphertext has a relative error bounded by

\[
\beta' = \beta_1 + \beta_2 + \beta_1 \beta_2 + \frac{B_{\text{mult}} + 2^{-p} \cdot B_{\text{scale}}}{M_1 M_2}
\]

from Lemmas 2 and 3. This relative error is close to \(\beta_1 + \beta_2\) 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 \(\beta_1\) and \(\beta_2\) the relative error after multiplication and rescaling is bounded by \(\beta_1 + \beta_2 + \beta^*\) for some fixed \(\beta^*\).

4 Homomorphic Evaluations of Matrix Operations

Along with the algorithms for homomorphic evaluation of approximate circuits, such as exponent, sigmoid, inverse, etc., described in Section 4 of [9], with the structure of MHEAAN we can also provide algorithms for homomorphic evaluation of approximate matrix multiplication and approximate matrix inverse.
4.1 Extracting Diagonal from a Matrix

One simple way to extract diagonal (or any other subset of values) from encrypted $n \times n$ matrix is described in Algorithm 1. We multiply the ciphertext by the identity matrix $I \in \mathbb{C}^{n \times n}$ component wisely, with the following rescaling procedure. Due to rescaling procedure this method cost us $p$ bits of a modulus.

Algorithm 1 Simple Diagonal Extraction

1: procedure SIMPLEDIAG($\text{ct}_A, \in \mathbb{R}_p^2, p$)
2: $\text{ct}_D \leftarrow \text{MHEAAN}.CMult(\text{ct}_A, I)$
3: $\text{MHEAAN}.ReScale(\text{ct}_D, p)$
4: return $\text{ct}_D$
5: end procedure

If we consider the following polynomial

$$i(X, Y) = X^{Mx \frac{N_x}{2} + \frac{N_x}{2}} \cdot (X^{N_x} + Y^{N_y}) \cdot (X^{N_x} + Y^{N_y}) \ldots (X^{N_x} + Y^{N_y})$$

we can easily check that $\phi_{n,n}^{-1}(i(X, Y)) = n \cdot I$ which means $i(X, Y)$ exactly encodes the identity matrix with $\log n$ precision bits: $i(X, Y) = \text{MHEAAN}.Encode(I, \log n)$. So we can extract a diagonal from an encrypted matrix multiplying by the polynomial $i(X, Y)$ and rescaling by $\log n$ bits. Normally in practice $p$ is notably larger than $\log n$.

Algorithm 2 Diagonal Extraction

1: procedure $\text{Diag}(\text{ct}_A, \in \mathbb{R}_p^2)$
2: $\text{ct}_D \leftarrow \text{MHEAAN}.PolyMult(\text{ct}_A, i)$
3: $\text{MHEAAN}.ReScale(\text{ct}_D, \log n)$
4: return $\text{ct}_D$
5: end procedure

By applying $\kappa_{0, N_y/2-k}$ to $i(X, Y)$ we obtain an encoding of a right shifted by $k$ bits identity matrix. We can then extract a right shifted by $k$ position diagonal from an encrypted matrix. We will call this procedure $\text{Diag}_k$, where $\text{Diag}_0$ is diagonal extraction. We will use this procedure in the following sections.

4.2 Matrix Multiplication

We adapt Fox matrix multiplication algorithm [18] to encrypted matrix multiplication. Let $\text{ct}_A, \text{ct}_B$ be encryptions of matrices $A, B \in \mathbb{C}^{n \times n}$ with power-of-two $n$. Let $I \in \mathbb{C}^{n \times n}$ be the identity matrix, and $\text{Diag}_k$ is procedure defined in previous section.
Algorithm 3 Matrix Multiplication

1: procedure MHEAAN.MatMult(ct_A, ct_B ∈ \mathbb{R}^2_ℓ, p)

2: \text{ct}_C ← 0

3: for \( k = 0 \) to \( n - 1 \) do

4: \text{ct}_{B_k} ← \text{Diag}_k(\text{ct}_B)

5: for \( j = 1 \) to \( \log(n) - 1 \) do

6: \text{ct}_{B_k} ← \text{MHEAAN}.Add(\text{ct}_{B_k}, \text{MHEAAN}.Rotate(\text{ct}_{B_k}, (0, 2^j)))

7: end for

8: \text{ct}_{A_k} ← \text{MHEAAN}.ModDown(\text{MHEAAN}.Rotate(\text{ct}_{A_k}, (\frac{N}{2^p} - k, 0)), \log n)

9: \text{ct}_{C_k} ← \text{MHEAAN}.Mult(\text{ct}_{A_k}, \text{ct}_{B_k})

10: \text{ct}_C ← \text{MHEAAN}.Add(\text{ct}_C, \text{ct}_{C_k})

11: end for

12: \text{ct}_C ← \text{MHEAAN}.ReScale(\text{ct}_C, p)

13: return \text{ct}_C

14: end procedure

Lemma 5 (Matrix Multiplication). Let \((\text{ct}_A, β_A \cdot 2^p, 2^p, ℓ)\) and \((\text{ct}_B, β_B \cdot 2^p, 2^p, ℓ)\) be encryptions of matrices \(A, B \in \mathbb{C}^{n \times n}\) respectively. The Algorithm 3 outputs \((\text{ct}_C, β_C \cdot n \cdot 2^p, n \cdot 2^p, ℓ - p - \log n)\) the valid encryption of \(C = AB\) where \(β_C = β_A + β_B + β^* + (\log n) \cdot β^*_n\).

Remark The plain matrix multiplication algorithm has complexity \(O(n^3)\). The Algorithm 3 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.

Remark The Algorithm 3 can be generalized to multiplication of matrices with arbitrary dimensions. We will omit the details as we need to consider many cases, although they are essentially similar.

4.3 Matrix Inverse

For matrix inverse we can adapt Schulz algorithm [36] to encrypted approximate inverse circuit. However for MHEAAN we use a matrix version algorithm described in [7] and adopted in [9] 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 \(A\) satisfies \(\|\hat{A}\| \leq δ < 1\) for \(\hat{A} = I - \frac{1}{t}A\), for some \(t \geq 0\) then we get

\[
\frac{1}{2^t}A(I + \hat{A})(I + \hat{A}^2)\ldots(I + \hat{A}^{2^{t-1}}) = 1 - \hat{A}^{2^t}
\]

We can see that \(\|\hat{A}^{2^t}\| \leq \|\hat{A}\|^{2^t} \leq δ^{2^t}\), hence \(\frac{1}{2^t} \prod_{j=0}^{t-1}(I + \hat{A}^{2^j}) = A^{-1}(1 - \hat{A}^{2^t})\) is an approximate inverse of \(A\) for \(δ^{2^t} \ll 1\). We will slightly strengthen the condition on \(δ\) in the following lemma:
Algorithm 4 Matrix Inverse

1: procedure MHEAAN.MatInv($c\bar{A} \in \mathbb{R}_c^{2 \ell, r, p \in \mathbb{Z}}$)
2:   $i = \text{MHEAAN.Encode}(I, p)$
3:   $ct_{A_0} \leftarrow ct_{\bar{A}}$
4:   $ct_{V_0} \leftarrow \text{MHEAAN.ModDown}(i + ct_{A}, p)$
5:   for $j = 0$ to $r - 1$
6:      $ct_{A_j} \leftarrow \text{MHEAAN.MatMult}(ct_{A_{j-1}}, ct_{A_{j-1}})$
7:      $ct_{V_{j+1}} \leftarrow \text{MHEAAN.MatMult}(ct_{V_j}, i + ct_{A_j})$
8:   end for
9:   $ct_{V_r} \leftarrow \text{MHEAAN.ReScale}(ct_{V_r}, t)$
10: return $ct_{V_r}$
11: end procedure

Lemma 6 (Matrix Inverse). Let $(ct_{\bar{A}}, \beta \cdot \delta 2^p/n, \delta 2^p/n, \ell)$ be an encryption of matrix $\bar{A} \in \mathbb{C}^{n \times n}$, and $\|A\| = \|I - \frac{1}{\ell} A\| \leq \delta < \frac{n-1}{n}$ for some $t$. The Algorithm 4 outputs $(ct_{V_r}, \beta_{V_r} \cdot n^{1/2} 2^p - t, n^{1/2} 2^p - t, \ell - (p + \log n) r - t)$ the valid encryption of $\bar{A}^{-1}$ where $\beta_{V_r} = 2\beta + \beta^* + (r + 1) \cdot (\beta^* + (\log n)\beta_s)$. So we have that the output message bound is close to $2^p - t$ and error growth linearly in $r$.

5 Implementation Results

In this section, we present parameters sets with experimental results. We also provide implementation results with concrete parameters. Our implementation is based on the NTL C++ library running over GMP. Every experimentation was performed on a machine with an Intel Core i5 running at 2.9 GHz processor using a parameter set with 80-bit security level. All computations are done in a single thread.

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, within an additional assumption that 2-RLWE sample with dimensions $N_x$ and $N_y$ is indistinguishable from a sample belonging to RLWE with $N = N_x N_y$ (Prop. 1 in [30]). In the implementation, we used the Gaussian distribution of standard deviation $\sigma = 3.2$ to sample error polynomials, and set $h = 64$ as the number of nonzero coefficients in a secret key $s(X, Y)$. We skip the results of evaluation a component wise power, inverse, exponent, sigmoid functions, etc. Please refer to [9] for more details on evaluating these circuits.

Evaluation of Matrix Circuits In Table 1, we present the parameter setting and performance results for matrix transposition, multiplication, a 16-th power, and inverse. Parameter $\gamma$ corresponds to the input precision bits and was calculated using Lemma 1. The average running times are only for ciphertext operations, excluding encryption and decryption procedures.

The homomorphic evaluation of the circuit $\text{M}^{16}$ for an $n \times n$ matrix $\text{M}$ is hard to be implemented in practice over the previous methods. Meanwhile, our
Table 1. Implementation results for $n \times n$ matrices $M$, $M_1$, $M_2$

<table>
<thead>
<tr>
<th>Function</th>
<th>Input params</th>
<th>HE params</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n$</td>
<td>$\gamma$</td>
<td>$N_x$</td>
</tr>
<tr>
<td>$M^T$</td>
<td>16</td>
<td>64</td>
<td>100</td>
</tr>
<tr>
<td>$M_1M_2$</td>
<td>16</td>
<td>17</td>
<td>$2^7$</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>100</td>
<td>30</td>
</tr>
<tr>
<td>$M^{16}$</td>
<td>16</td>
<td>64</td>
<td>$2^7$</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>30</td>
<td>3.9min</td>
</tr>
<tr>
<td>$M^{-1}$</td>
<td>16</td>
<td>12</td>
<td>310</td>
</tr>
</tbody>
</table>

scheme can compute this circuit by squaring a matrix 4 times and it takes about 4 minutes for a $64 \times 64$ matrix. Computing the matrix inverse homomorphically is done by evaluating a matrix polynomial up to degree 15 as was shown in Algorithm 4.

6 Conclusion

In this work, we presented MHEAAN scheme - a multidimensional variant of HEAAN homomorphic encryption scheme which supports an approximate arithmetics over ciphertexts. MHEAAN scheme takes advantage of the HEAAN scheme and provides a tensor packing method in a single ciphertext. We finally constructed a practical HE that supports standard HE operations as well as operations with matrices.

We believe that the idea of multidimensional variants could be applied to other existing schemes, with a great potential to homomorphic computations on matrices and tensors.

References


A Noise Estimations

We follow the heuristic approach in [20] generalizing it to two variable polynomials. Assume that a two variable polynomial \( a(X, Y) \in \mathcal{R} = \mathbb{Z}[X, Y]/(\Phi_M(X), \Phi_M(Y)) \)
sampled from one of above distributions, so its nonzero entries are independently and identically distributed. The value $a(\xi_{M_z}, \xi_{M_y})$ can be obtained by first computing $N_z$ inner products of vectors of coefficients of $a$ corresponding to a power $Y^j$ for $j = 0, \ldots, N_y - 1$ by a fixed vector $(1, \xi_{M_z}, \ldots, \xi_{N_z}^{N_y})$ of Euclidean norm $\sqrt{N_z}$ and then computing inner product of the results by a fixed vector $(1, \xi_{M_y}, \ldots, \xi_{N_y}^{N_y})$ of Euclidean norm $\sqrt{N_y}$. Then $a(\xi_{M_z}, \xi_{M_y})$ has variance $V = \sigma^2 N_z N_y = \sigma^2 N$, where $\sigma^2$ is the variance of each coefficient of $a$. Hence $a(\xi_{M_z}, \xi_{M_y})$ has the variances $V_U = 2^j N/12$, $V_G = \sigma^2 N$ and $V_Z = \rho N$, when $a$ is sampled from $R_\ell$, $DG(\sigma^2)$, $ZO(\rho)$ respectively. In particular, $a(\xi_{M_z}, \xi_{M_y})$ has the variance $V_H = h$ when $a(X)$ is chosen from $HW(h)$. Moreover, we can assume that $a(\xi_{M_z}, \xi_{M_y})$ 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 roots of unity $(\xi_{M_z}, \xi_{M_y})$ share the same variance. Hence, we will use $6\sigma$ as a high-probability bound on the canonical embedding norm of a when each coefficient has a variance $\sigma^2$. For a multiplication of two independent random variables close to Gaussian distributions with variances $\sigma_1^2$ and $\sigma_2^2$, we will use a high-probability bound $16\sigma_1\sigma_2$.

**Proof of Lemma 1.**

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

$$\|\langle ct, sk \rangle - m \pmod{2^L}\|_\infty^{\text{mcan}} = \|v \cdot e + e_1 + e_0 \cdot s\|_\infty^{\text{mcan}}$$

$$\leq \|v \cdot e\|_\infty^{\text{mcan}} + \|e_1\|_\infty^{\text{mcan}} + \|e_0 \cdot s\|_\infty^{\text{mcan}}$$

$$\leq 8\sqrt{2} \cdot \sigma N + 6\sqrt{N} + 16\sqrt{hN}.$$  

\qed

**Proof of Lemma 2.**

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

We may assume that each coefficient of $\tau_0$ and $\tau_1$ 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

$$\|e_{\text{scale}}\|_\infty^{\text{mcan}} \leq \|\tau_0 \cdot s\|_\infty^{\text{mcan}} + \|\tau_1\|_\infty^{\text{mcan}} \leq 6\sqrt{N/12} + 16\sqrt{hN/12}$$

as desired. \qed

**Proof of Lemma 3.**
Proof. Let \( \text{ct}_i = (a_i, b_i) \) for \( i = 1, 2 \). Then \( \langle \text{ct}_i, \text{sk} \rangle = m_i + e_i \pmod{2^t} \) for some polynomials \( e_i \in S \) such that \( \|e_i\|_{\text{mean}} \leq B_i \). 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 2 that the ciphertext \( \text{ct}_{\text{mult}} \leftarrow (d_1, d_2) + \lfloor 2^{-L} \cdot (d_0 \cdot evk \pmod{2^{t+L}}) \rfloor \) contains an additional error \( e'' = 2^{-L} \cdot d_0 e' \) and a rounding error bounded by \( B_{\text{scale}} \). We may assume that \( d_0 \) behaves as a uniform random variable on \( \mathcal{R}_e \), so \( 2^t \| e'' \|_{\text{can}} \) is bounded by \( 16 \sqrt{Nq_e} / 12 \sqrt{N} \sigma^2 = 8Nq_e / \sqrt{3} = B_{\text{ks}} \cdot 2^t \). Therefore, \( \text{ct}_{\text{mult}} \) is an encryption of \( m_1 \cdot m_2 \) with an error bounded by

\[
\|m_1 e_2 + m_2 e_1 + e_1 e_2 + e''\|_{\infty} + B_{\text{scale}} \leq M_1 B_2 + M_2 B_1 + B_1 B_2 + 2^{-L} \cdot 2^t \cdot B_{\text{ks}} + B_{\text{scale}}
\]

as desired. \( \square \)

**Proof of Lemma 4.**

Proof. Let prove the lemma for conjugation, proofs of others are the same. The vector \( (a', b') = (\kappa_{-1,1}(a), \kappa_{-1,1}(b)) \pmod{2^t} \) can be viewed as an encryption of \( \mathbb{Z} \) with and error \( \kappa_{-1,1}(e) \) with respect to the secret vector \( (\kappa_{-1,1}(s), 1) \). Using proof of Lemma 3 we can get that \( \text{ct}_{\text{ej}} \) is an encryption of \( \hat{Z} \) with an error bounded by

\[
\|\kappa_{-1,1}(e) + e''\|_{\text{can}} + B_{\text{scale}} \leq B + 2^{-L} \cdot 2^t \cdot B_{\text{ks}} + B_{\text{scale}}
\]

as desired. \( \square \)

**Proof of Lemma 5.**

Proof. From Lemma 4 and the following remark about the relative error we can see that bound of message increment only after summations in line 10 of Algorithm 3, 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 \( \beta_s \) after rotation and increases by \( \beta_\star \) after multiplication. So the relative error of each summand in line 10 is bounded by \( \beta_A + \beta_B + \beta_\star + (\log n) \cdot \beta_s \). \( \square \)

**Proof of Lemma 6.**

Proof. From Lemma 5 the message of \( \text{ct}_{A_j} \) is bounded by \( \delta^{2^t} 2^p / n \) which implies that the message of \( \text{ct}_{V_\star} \) is bounded by

\[
2^{p-t} \prod_{j=0}^{r-1} (1 + \delta^{2^j} / n) < \frac{2^{p-t}}{(1 - \delta)^{1/n}} < n^{1/n} 2^{p-t}
\]
The relative error $\beta_j$ of $c_{tA_j}$ is bounded by $\beta_j \leq 2^j(\beta + \beta^* + (\log n)\beta_*)$, which implies that the relative error $\beta'_j$ of $c_{tA_j} + i$ is bounded by

$$\beta'_j \leq \beta_j / \left(1 + \frac{n}{\delta^*}\right)$$

Using induction on $j$, we can show that a relative error $\beta''_j$ of $c_{tV_j}$ is bounded by

$$\beta''_j \leq \left(\sum_{k=0}^{j-1} \frac{2^k \delta^{2k}}{n + \delta^{2k}}\right) \cdot (\beta + \beta^* + (\log n)\beta_*) + (j - 1) \cdot (\beta^* + (\log n)\beta_*) \leq$$

$$\frac{1}{n} \sum_{k=0}^{j-1} (2^k \delta^{2k}) \cdot (\beta + \beta^* + (\log n)\beta_*) + (j - 1) \cdot (\beta^* + (\log n)\beta_*) \leq$$

$$\frac{2}{n(1 - \delta)} \cdot (\beta + \beta^* + (\log n)\beta_*) + (j - 1) \cdot (\beta^* + (\log n)\beta_*) \leq$$

$$2\beta + (j + 1) \cdot (\beta^* + (\log n)\beta_*)$$

Finally after last rescaling we get the required result $\square$