A New Framework for Fast Homomorphic Matrix Multiplication[⋆]

Xiaopeng Zheng[1,2], Hongbo Li[1,2], and Dingkang Wang[1,2]

[1] KLMM, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China. {hli,dwang}@mmrc.iss.ac.cn, zhengxiaopeng@amss.ac.cn.
[2] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China.

Abstract. Homomorphic Encryption (HE) is one of the mainstream cryptographic tools used to enable secure outsourced computation. A typical task is secure matrix computation. Popular HE schemes are all based on the problem of Ring Learning with Errors (RLWE), where the messages are encrypted in a ring. In general, the ring dimension should be large to ensure security, which is often larger than the matrix size. Hence, exploiting the ring structure to make fast homomorphic matrix computation has been an important topic in HE.
In this paper, we present a new framework for encoding a matrix and performing multiplication on encrypted matrices. The new framework requires fewer basic homomorphic operations for matrix multiplication. Suppose that the ring dimension is n and the matrix size is $d \times d$ with $d = n^\rho$. (1) In the compact case where $\rho \leq \frac{1}{3}$, the multiplication of two encrypted matrices requires $\tilde{O}(1)$ basic homomorphic operations, which include plaintext-ciphertext multiplications, ciphertext-ciphertext multiplications, and homomorphic Galois automorphisms. (2) In the large sized case where $\rho > \frac{1}{3}$, our new method requires $O\big(d^{(1-\frac{1}{3\rho})\cdot\log_2 7}\big)$ basic homomorphic operations, which is better than all existing methods.
In addition, the new framework reduces the communication cost, since it requires fewer key-switching keys. The number of key-switching keys is reduced from $O(d)$ to $O(\log d)$.

Keywords: Homomorphic Encryption · Secure Outsourced Matrix Multiplication · Tensor Ring · Lattice Basis and Dual Basis · Galois Automorphism.

1 Introduction

Fully homomorphic encryption (FHE) is a revolutionary cryptographic technique that enables computations to be performed on encrypted data without the need

[⋆] Supported partially by National Key Research and Development Project (2020YFA0712300) and the National Science Foundation of China under Grant No. 12171469.

for decryption. With homomorphic encryption, data can be securely outsourced to cloud service providers or processed by third parties, without compromising confidentiality.

In 2009, Gentry launched a pioneering work on fully homomorphic encryption (FHE) [9]. In 2014, Brakerski et al. [1] proposed a more practical homomorphic encryption scheme that supports finite homomorphic operations without decryption. Since then, extensive research has been conducted to enhance the efficiency of FHE, such as those in Cheon et al. [3], Chillotti et al. [5], Ducas et al. [7], Gentry [10], Lee et al. [18], and Liu et al. [19].

Secure Matrix Computation based on HE. Secure matrix multiplication is a trending research topic due to its importance in secure data analysis and machine learning. A naive approach to securely multiply two matrices of size $d \times d$ is to use d^2 distinct ciphertexts to represent each input matrix. The method is very inefficient and requires a large amount of communication. Therefore, recently, more efficient algorithms have been proposed. Some methods are listed as follows:

– In 2014, Halevi et al. [12] conducted research on secure matrix-vector multiplication, where both the matrix and the vector are encrypted. Lu et al. [20] and Wang et al. [27] extended the matrix-vector multiplication to matrix-matrix multiplication for two encrypted matrices. However, their approach requires d ciphertexts to represent a $d \times d$ matrix, and requires $O(d^2)$ plaintext-ciphertext multiplications, $O(d^2)$ ciphertext-ciphertext multiplications, and $O(d^2 \log d)$ homomorphic automorphisms to compute the multiplication of two $d \times d$ matrices, see Table 1.
– In 2018, Cheon et al. [4] proposed a more efficient method for secure matrix multiplication, using only one ciphertext to represent a matrix. They extract the diagonals from one matrix and rotate another matrix. As a consequence, this method requires a total of $O(d)$ plaintext-ciphertext multiplications, $O(d \log d)$ homomorphic automorphisms, and $O(d)$ ciphertext-ciphertext multiplications to compute the multiplication of two encrypted matrices. Jiang et al. [17] also employed the use of a single ciphertext to represent a matrix, and then introduced a new approach to compute matrix multiplication, which reduced the number of homomorphic automorphisms from $O(d \log d)$ to $O(d)$. The depth of their approach consists of a single ciphertext-ciphertext multiplication and two plaintext-ciphertext multiplications.
– In 2020, Chen et al. [2] combined the algorithm of HE matrix multiplication in [17] with SPDZ framework to reduce the communication cost.
– In 2022, Jang et al. [16] improved upon the algorithm presented by Jiang et al. [17]. The enhanced algorithm achieves a depth of one ciphertext-ciphertext multiplication and one plaintext-ciphertext multiplication. Recently, Rizomiliotis and Triakosia [25] completed the work started by Jang et al. [17] by reducing the number of ciphertext-ciphertext multiplications

from $O(d)$ to $O(1)$. However, it still requires a total of $O(d)$ homomorphic automorphisms.

– In 2018-2023, several studies in Jiang et al. [17], Huang et al. [15], and Huang et al. [14], investigate techniques for secure non-square matrix multiplication.

Table 1 presents a summary of the number of basic homomorphic operations of these algorithms.

Table 1. Number of basic homomorphic operations for one matrix multplication

Methodology	Maximum value of d	CMult[a]	Aut[b]	Mult[c]	Required Depth for one matrix multiplication
[20,27]	$O(n)$	$O(d^2)$	$O(d^2 \log d)$	$O(d^2)$	1 CMult + 1 Mult
[4,23]	$O(n^{\frac{1}{2}})$	$O(d)$	$O(d \log d)$	$O(d)$	1 CMult + 1 Mult
[17]	$O(n^{\frac{1}{2}})$	$O(d)$	$O(d)$	$O(d)$	2 CMult + 1 Mult
[16]	$O(n^{\frac{1}{2}})$	$O(d)$	$O(d)$	$O(d)$	1 CMult + 1 Mult
[25]	$O(n^{\frac{1}{3}})$	$O(d)$	$O(d)$	$O(1)$	1 CMult + 1 Mult
$\text{Ours}_{\text{comp}}$	$O(n^{\frac{1}{3}})$	$O(1)$	$O(\log d)$	$O(1)$	1 CMult + 1 Mult
$\text{Ours}_{\text{large}}$	Infinity[d]	$O(d^{2-\frac{2}{3\rho}})$	$\tilde{O}(d^{2-\frac{2}{3\rho}})$	$O\left(d^{\left(1-\frac{1}{3\rho}\right)\log_2 7}\right)$	1 CMult + 1 Mult

[a] CMult: plaintext-ciphertext multiplication;
[b] Aut: homomorphic Galois automorphism;
[c] Mult: ciphertext-ciphertext multiplication.
[d] $d > n^{\frac{1}{3}}$ but no restriction of the upper bound.

In Table 1, n is the ring dimension and $d = n^{\rho}$. The estimated numbers of basic homomorphic operations only works when d is not larger than the maximum value, otherwise, one needs to partition a matrix into several sub-matrices and encrypt each one separately, and then use block matrix multiplication.

1.1 Our Results

The current best-performing HE schemes rely on the hardness of the Learning with Errors (LWE) problem or it ring variant (RLWE) [21,24]. In RLWE-based HE schemes, the plaintext space is $\mathcal{R}_p = \mathbb{Z}_p[x]/(\Phi_m(x))$, where $\Phi_m(x)$ is the m-th cyclotomic polynomial. Let $n = \deg(\Phi_m(x))$. In general, for security consideration, the value of n should not be small.

In this paper, we present a new framework for homomorphic matrix multiplication. It outperforms all other existing methods based on homomorphic encryption (HE). The main result is as follows:

Suppose that the size of the square matrix is $d \times d$, where $d = n^{\rho}$.

Theorem 1.1 (Main Result). *In the compact case where $\rho \le 1/3$, the homomorphic matrix multiplication requires $O(1)$ ciphertext-ciphertext multiplications and $O(\log d)$ homomorphic automorphisms. In the large sized case where $\rho > 1/3$, the homomorphic matrix multiplication requires $O\left(d^{\left(1-\frac{1}{3\rho}\right)\log_2 7}\right)$ ciphertext-ciphertext multiplications and $O(d^{2-\frac{2}{2\rho}} \log d)$ homomorphic automorphisms.*

In comparison with other methods in Table 1, if $\rho \leq 1/2$, it is easy to deduce that $\left(1 - \frac{1}{3\rho}\right) \log_2 7 < 1$ and $2 - \frac{2}{3\rho} < 1$, so our method is better than the methods in [4,17,23,16]. If $\rho \leq 1$, then by $\left(1 - \frac{1}{3\rho}\right) \log_2 7 < 2$ and $2 - \frac{2}{2\rho} < 2$, our method is better than the methods in [20,27]. If $\rho \leq \frac{1}{3}$, our method only requires $O(\log d)$ homomorphic automorphisms and $O(1)$ plaintext-ciphertext multiplications, which is less than that in [25].

As for communication, as our framework requires fewer homomorphic automorphisms, it demands less key-switching keys. This reduces the communication load when compared with [4,16,17,23,25]. Indeed, in [4,16,17,23,25], the number of key-switching keys is $O(d)$, while in our method, the number of key-switching keys is only $O(\log d)$.

1.2 Our Techniques

Let $\mathcal{R} = \mathbb{Z}[x]/(\Phi_m(x))$, where $\Phi_m(x)$ is the m-th cyclotomic polynomial with $\deg(\Phi_m(x)) = n$. In our new framework, we choose $m = m_1 m_2 m_3$, with m_1, m_2 and m_3 pairwise coprime. Let $\mathcal{R}_i = \mathbb{Z}[x]/(\Phi_{m_i}(x))$, with $i = 1, 2, 3$. According to the classical theory of algebraic number theory [19,22], we have $\mathcal{R} \cong \mathcal{R}_1 \otimes \mathcal{R}_2 \otimes \mathcal{R}_3$. Let $\{\mathbf{u}_1, \ldots, \mathbf{u}_r\}$, $\{\mathbf{v}_1, \ldots, \mathbf{v}_s\}$ and $\{\mathbf{w}_1, \ldots, \mathbf{w}_t\}$ be $\mathbb{Z}$-bases of $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$, respectively, and let $\{\mathbf{u}_1^\vee, \ldots, \mathbf{u}_r^\vee\}$, $\{\mathbf{v}_1^\vee, \ldots, \mathbf{v}_s^\vee\}$ and $\{\mathbf{w}_1^\vee, \ldots, \mathbf{w}_t^\vee\}$ be the corresponding dual bases in $\mathcal{R}_1^\vee$, $\mathcal{R}_2^\vee$ and $\mathcal{R}_3^\vee$ respectively, where $n = r \cdot s \cdot t$. We choose m_1, m_2 and m_2 properly, such that $r \approx s \approx t$ asymptotically.

Let

$$d_0 = \min\{r, s, t\}, \tag{1.1}$$

then $d_0 \approx n^{\frac{1}{3}}$. In the compact sized case where $d \leq d_0$, suppose that $A = (a_{ij})$ and $B = (b_{ij})$ are two integer matrices of size $d \times d$. Inspired by [19], we find that the matrix multiplication can be achieved by polynomial multiplication and trace operation. Specifically, let

$$\mathbf{m}_1 = \sum_{i=1}^{d} \sum_{j=1}^{d} a_{ij} \mathbf{u}_i \mathbf{w}_j \qquad \text{and} \qquad \mathbf{m}_2 = \sum_{i=1}^{d} \sum_{j=1}^{d} b_{ij} \mathbf{w}_i^\vee \mathbf{v}_j. \tag{1.2}$$

Then we have

$$\mathrm{Tr}(\mathbf{m}_1 \cdot \mathbf{m}_2) = \sum_{i=1}^{d} \sum_{j=1}^{d} c_{ij} \mathbf{u}_i \mathbf{v}_j, \tag{1.3}$$

with $(c_{ij})_{1 \leq i,j \leq d} = A \cdot B$ (see Theorem 3.2). This surprising discovery allows us to implement matrix multiplication through polynomial operations.

However, if A and B are encoded as (1.2), we cannot directly implement matrix addition through polynomial addition. Therefore, we hope to encode matrices A and B using the same method. Specifically, we hope that A and B are encoded as

$$\mathbf{m}_1 = \sum_{i=1}^{d} \sum_{j=1}^{d} a_{ij} \mathbf{u}_i \mathbf{v}_j \qquad \text{and} \qquad \mathbf{m}_2 = \sum_{i=1}^{d} \sum_{j=1}^{d} b_{ij} \mathbf{u}_i \mathbf{v}_j. \tag{1.4}$$

Then we can obtain

$$\mathbf{m}_1 + \mathbf{m}_2 = \sum_{i=1}^{d}\sum_{j=1}^{d}(a_{ij} + b_{ij})\mathbf{u}_i\mathbf{v}_j. \tag{1.5}$$

To bridge the gap between the encoding methods (1.2) and (1.4), we propose a homomorphic basis switching technique in Section 3.3. We summarize the new framework homomorphic matrix multiplication for the compact sized case in Figure 1.

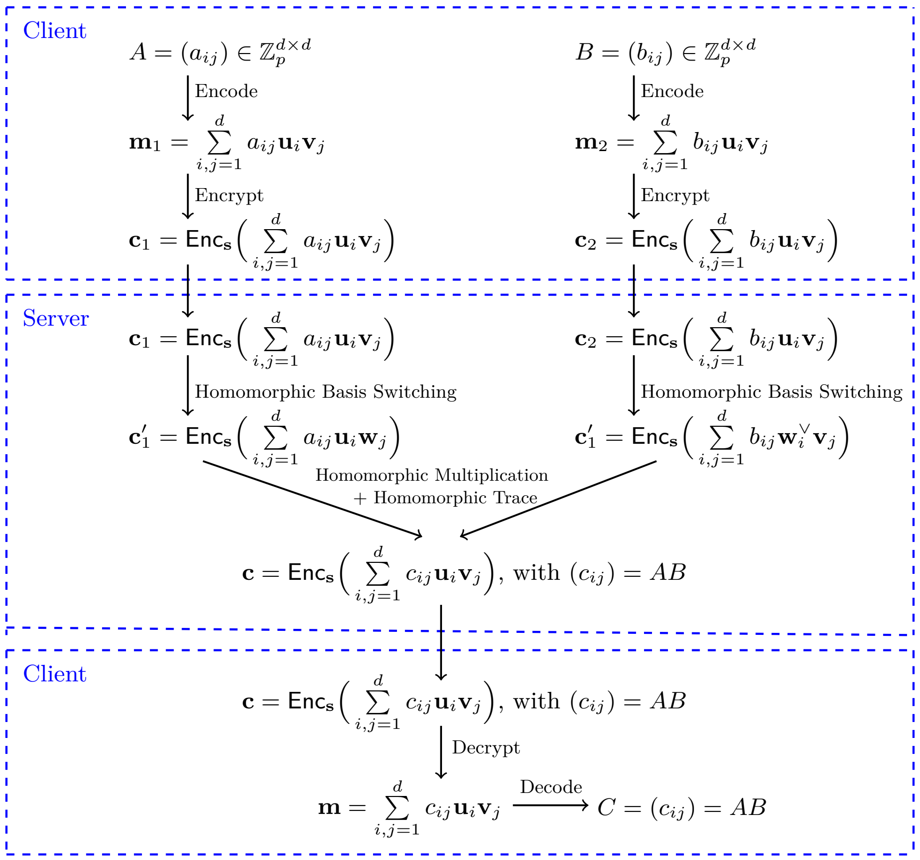
Client
$A = (a_{ij}) \in \mathbb{Z}_p^{d\times d}$
Encode
$\mathbf{m}_1 = \sum_{i,j=1}^{d} a_{ij}\mathbf{u}_i\mathbf{v}_j$
Encrypt
$\mathbf{c}_1 = \mathsf{Enc}_{\mathbf{s}}\left(\sum_{i,j=1}^{d} a_{ij}\mathbf{u}_i\mathbf{v}_j\right)$
$B = (b_{ij}) \in \mathbb{Z}_p^{d\times d}$
Encode
$\mathbf{m}_2 = \sum_{i,j=1}^{d} b_{ij}\mathbf{u}_i\mathbf{v}_j$
Encrypt
$\mathbf{c}_2 = \mathsf{Enc}_{\mathbf{s}}\left(\sum_{i,j=1}^{d} b_{ij}\mathbf{u}_i\mathbf{v}_j\right)$
Server
$\mathbf{c}_1 = \mathsf{Enc}_{\mathbf{s}}\left(\sum_{i,j=1}^{d} a_{ij}\mathbf{u}_i\mathbf{v}_j\right)$
Homomorphic Basis Switching
$\mathbf{c}_1' = \mathsf{Enc}_{\mathbf{s}}\left(\sum_{i,j=1}^{d} a_{ij}\mathbf{u}_i\mathbf{w}_j\right)$
$\mathbf{c}_2 = \mathsf{Enc}_{\mathbf{s}}\left(\sum_{i,j=1}^{d} b_{ij}\mathbf{u}_i\mathbf{v}_j\right)$
Homomorphic Basis Switching
$\mathbf{c}_1' = \mathsf{Enc}_{\mathbf{s}}\left(\sum_{i,j=1}^{d} b_{ij}\mathbf{w}_i^{\vee}\mathbf{v}_j\right)$
Homomorphic Multiplication
+ Homomorphic Trace
$\mathbf{c} = \mathsf{Enc}_{\mathbf{s}}\left(\sum_{i,j=1}^{d} c_{ij}\mathbf{u}_i\mathbf{v}_j\right)$, with $(c_{ij}) = AB$
Client
$\mathbf{c} = \mathsf{Enc}_{\mathbf{s}}\left(\sum_{i,j=1}^{d} c_{ij}\mathbf{u}_i\mathbf{v}_j\right)$, with $(c_{ij}) = AB$
Decrypt
$\mathbf{m} = \sum_{i,j=1}^{d} c_{ij}\mathbf{u}_i\mathbf{v}_j \xrightarrow{\text{Decode}} C = (c_{ij}) = AB$

Fig. 1. (Informal) Homomorphic Matrix Multiplication for the compact sized case.

If $d > d_0 = \min\{r, s, t\}$, we divide the matrix into $L \times L$ blocks, and then use block matrix multiplication (Theorem 3.4), where each block is a $d_0 \times d_0$ matrix. However, if we encrypt each matrix directly, then a total of L^2 matrices of size $d_0 \times d_0$ will be encrypted. This will lead to an increase in both the encryption and the decryption time, as well as the cost of communication. We propose a new technique to pack these matrices into one plaintext polynomial. The main idea is to use the basis $\{\mathbf{w}_1^\vee, \ldots, \mathbf{w}_t^\vee\}$ for packing these L^2 matrices. Now that a maximum of t matrices can be packed with this basis, L can be at most $\lfloor t^{\frac{1}{2}} \rfloor$, so the maximal size of a matrix that can be packed in a plaintext is $d \times d$, with $d = \lfloor t^{\frac{1}{2}} \rfloor \cdot d_0$. Since t and d_0 are both approximately equal to $n^{\frac{1}{3}}$, then $d = \lfloor t^{\frac{1}{2}} \rfloor \cdot d_0 = O(n^{\frac{1}{2}})$.

The procedure of large sized case for $d > d_0$ in the illustrative case $L = 2$ is shown in Figure 2. In Figure 2, the homomorphic unpacking process only needs to be done once when receiving the ciphertexts, and the homomorphic packing process only needs to be done once when sending the ciphertexts. For block matrix multiplication, we apply Strassen algorithm, which can reduce the number of ciphertext multiplications from $O(L^3)$ to $O(L^{\log_2 7})$.

1.3 Organization

The rest of the paper is organized as follows. In Section 2, we provide the necessary background on algebraic number theory and BGV homomorphic encryption scheme. In Section 3, we present a new framework for plaintext matrix multiplication via tensor ring. In Section 4, we extend the framework in Section 3 to the homomorphic case, and present a homomorphic packing and unpacking technique to reduce the communication cost. Section 5 provides more detailed comparisons with existing works under specific parameters. Section 6 presents details of implementation and a discussion of the experimental results. Section 7 provides the conclusions and future work.

2 Preliminaries

Notations. Let $\mathbb{Z}$ denote the set of integers, $\mathbb{Q}$ denote the set of rational numbers. Notation log refers to the base-2 logarithm. For a positive $k \in \mathbb{Z}$, let $[k]$ be the set of integers $\{0, ..., k-1\}$.

For $m \in \mathbb{N}$, let $\varphi(m)$ denote Euler's totient function. Denote by $K = \mathbb{Q}[x]/(\Phi_m(x))$ the m-th cyclotomic field, $\mathcal{R} = \mathbb{Z}[x]/(\Phi_m(x))$ the ring of integers of K, and $\mathcal{R}_Q = \mathbb{Z}_Q[x]/(\Phi_m(x))$ the residue ring of $\mathcal{R}$ modulo Q. Elements of the ring $\mathcal{R}$ or $\mathcal{R}_Q$ will be denoted in lowercase bold, e.g. $\mathbf{a} \in R$. The coefficients of an element $\mathbf{a} \in R$ will be denoted by a_i, i.e. $\mathbf{a} = \sum_{i=0}^{d-1} a_i \cdot x^i$. We use $\mathbb{Z} \cap (-Q/2, Q/2]$ as a representative of $\mathbb{Z}_Q$ for integer a, and denote by $[a]_Q$ the reduction of an integer a modulo Q.

2.1 Algebraic Number Theory Background

We present some necessary background of algebraic number theory. Further details can be found in reference [22,19].

$$\begin{tikzcd}[column sep=small]
\text{Client} & & & \\
& {\begin{array}{c} A = \begin{pmatrix} A^{(1)} & A^{(2)} \\ A^{(3)} & A^{(4)} \end{pmatrix}, \\ \text{with } A^{(k)} = (a_{ij}^{(k)}) \in \mathbb{Z}_p^{d_0 \times d_0} \end{array}} \arrow[d, "\text{Encode}"] & & {\begin{array}{c} B = \begin{pmatrix} B^{(1)} & B^{(2)} \\ B^{(3)} & B^{(4)} \end{pmatrix}, \\ \text{with } B^{(k)} = (b_{ij}^{(k)}) \in \mathbb{Z}_p^{d_0 \times d_0} \end{array}} \arrow[d, "\text{Encode}"] \\
& \mathbf{m}_1 = \sum_{k=1}^{4} \Big(\sum_{i,j=1}^{d_0} a_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j \Big) \mathbf{w}_k^{\vee} \arrow[d, "\text{Encrypt}"] & & \mathbf{m}_2 = \sum_{k=1}^{4} \Big(\sum_{i,j=1}^{d_0} b_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j \Big) \mathbf{w}_k^{\vee} \arrow[d, "\text{Encrypt}"] \\
& \mathbf{c}_1 = \mathsf{Enc}_{\mathbf{s}}(\mathbf{m}_1) \arrow[d] & & \mathbf{c}_2 = \mathsf{Enc}_{\mathbf{s}}(\mathbf{m}_2) \arrow[d] \\
\text{Server} & & & \\
& \mathbf{c}_1 = \mathsf{Enc}_{\mathbf{s}}(\mathbf{m}_1) \arrow[d, "\text{Homomorphic Blockwise Unpacking}"] & & \mathbf{c}_2 = \mathsf{Enc}_{\mathbf{s}}(\mathbf{m}_2) \arrow[d, "\text{Homomorphic Blockwise Unpacking}"] \\
& \begin{pmatrix} \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} a_{ij}^{(1)} \mathbf{u}_i \mathbf{v}_j\Big) & \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} a_{ij}^{(2)} \mathbf{u}_i \mathbf{v}_j\Big) \\ \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} a_{ij}^{(3)} \mathbf{u}_i \mathbf{v}_j\Big) & \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} a_{ij}^{(4)} \mathbf{u}_i \mathbf{v}_j\Big) \end{pmatrix} \arrow[d, "\text{Homomorphic Basis Switching}"] & & \begin{pmatrix} \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} b_{ij}^{(1)} \mathbf{u}_i \mathbf{v}_j\Big) & \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} b_{ij}^{(2)} \mathbf{u}_i \mathbf{v}_j\Big) \\ \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} b_{ij}^{(3)} \mathbf{u}_i \mathbf{v}_j\Big) & \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} b_{ij}^{(4)} \mathbf{u}_i \mathbf{v}_j\Big) \end{pmatrix} \arrow[d, "\text{Homomorphic Basis Switching}"] \\
& \begin{pmatrix} \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} a_{ij}^{(1)} \mathbf{u}_i \mathbf{w}_j\Big) & \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} a_{ij}^{(2)} \mathbf{u}_i \mathbf{w}_j\Big) \\ \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} a_{ij}^{(3)} \mathbf{u}_i \mathbf{w}_j\Big) & \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} a_{ij}^{(4)} \mathbf{u}_i \mathbf{w}_j\Big) \end{pmatrix} \arrow[dr] & {\begin{array}{c} \text{Block Homomorphic Multiplication} \\ \text{Homomorphic Trace} \end{array}} & \begin{pmatrix} \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} b_{ij}^{(1)} \mathbf{w}_i^{\vee} \mathbf{v}_j\Big) & \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} b_{ij}^{(2)} \mathbf{w}_i^{\vee} \mathbf{v}_j\Big) \\ \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} b_{ij}^{(3)} \mathbf{w}_i^{\vee} \mathbf{v}_j\Big) & \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} b_{ij}^{(4)} \mathbf{w}_i^{\vee} \mathbf{v}_j\Big) \end{pmatrix} \arrow[dl] \\
& & \begin{pmatrix} \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} c_{ij}^{(1)} \mathbf{u}_i \mathbf{v}_j\Big) & \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} c_{ij}^{(2)} \mathbf{u}_i \mathbf{v}_j\Big) \\ \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} c_{ij}^{(3)} \mathbf{u}_i \mathbf{v}_j\Big) & \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{i,j} c_{ij}^{(4)} \mathbf{u}_i \mathbf{v}_j\Big) \end{pmatrix} \text{ with } C = \begin{pmatrix} C^{(1)} & C^{(2)} \\ C^{(3)} & C^{(4)} \end{pmatrix} \text{ and } C^{(k)} = (c_{ij}^{(k)}). \arrow[d, "\text{Homomorphic Blockwise Packing}"] & \\
& & \mathbf{c} = \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{k=1}^{4} \Big(\sum_{i,j=1}^{d_0} c_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j \Big) \mathbf{w}_k \Big) \arrow[d] & \\
\text{Client} & & & \\
& & \mathbf{c} = \mathsf{Enc}_{\mathbf{s}}\Big(\sum_{k=1}^{4} \Big(\sum_{i,j=1}^{d_0} c_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j \Big) \mathbf{w}_k \Big) \arrow[d, "\text{Decrypt}"] & \\
& & \mathbf{m} = \sum_{k=1}^{4} \Big(\sum_{i,j=1}^{d} c_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j \Big) \mathbf{w}_k \xrightarrow{\text{Decode}} C = \begin{pmatrix} C^{(1)} & C^{(2)} \\ C^{(3)} & C^{(4)} \end{pmatrix} = AB &
\end{tikzcd}$$

Fig. 2. (Informal) Homomorphic matrix multiplication for the illustrative example $L = 2$.

Number Fields. Number fields are field extensions expressed as $K = \mathbb{Q}(\alpha)$ by adjoining an α to $\mathbb{Q}$, where α is a root of an irreducible polynomial $f(x)$ in $\mathbb{Z}[x]$. Let ξ_m represent the m-th root of unity. The field $\mathbb{Q}(\xi_m)$ is known as the m-th cyclotomic field, which is isomorphic to $\mathbb{Q}[x]/(\Phi_m(x))$. The Galois group of $\mathbb{Q}(\xi_m)$ over $\mathbb{Q}$ is denoted by $\mathrm{Gal}(\mathbb{Q}(\xi_m)/\mathbb{Q})$. It is well-known that this Galois group is isomorphic to the multiplicative group $\mathbb{Z}_m^*$ consisting of invertible residues modulo m.

Ring of Integers. An *algebraic integer* is an algebraic number whose minimal polynomial over the rationals has integer coefficients. Denote the subset of algebraic integers in the number field K by $\mathcal{O}_K$. It forms a ring known as the *ring of integers* of K.

Trace. Let K be a Galois extension over k. The trace $\mathrm{Tr}_{K/k}(\mathbf{a})$ of an element $\mathbf{a} \in K$ is defined as the sum of its embeddings:

$$\mathrm{Tr}_{K/k}(\mathbf{a}) = \sum_{\sigma \in \mathrm{Gal}(K/k)} \sigma(\mathbf{a}). \tag{2.1}$$

Duality. Let $K = \mathbb{Q}[x]/(\Phi_m(x))$ and $\mathcal{R} = \mathbb{Z}[x]/(\Phi_m(x))$. The *dual* of $\mathcal{R}$ is defined as:

$$\mathcal{R}^\vee = \{\mathbf{a} \in K : \mathrm{Tr}_{K/\mathbb{Q}}(\mathbf{a}\mathcal{R}) \subset \mathbb{Z}\}. \tag{2.2}$$

For any $\mathbb{Z}$-basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ of $\mathcal{R}$, its *dual basis* is denoted as $\mathbf{B}^\vee = \{\mathbf{b}_1^\vee, \ldots, \mathbf{b}_n^\vee\} \subset \mathcal{R}^\vee$, which is characterized by

$$\mathrm{Tr}_{K/\mathbb{Q}}\left(\mathbf{b}_i \cdot \mathbf{b}_j^\vee\right) = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

A method to compute a $\mathbb{Z}$-basis of $\mathcal{R}$ and its corresponding dual basis (referred to as decoding basis) can be found in [22, Lemma 6.3].

Lemma 2.1 ([22]). *Let m be a power of a prime p and let $m' = m/p$. Let $n = \varphi(m)$. Then*

1. $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} = \{x^{-j} : j = 0, \ldots, n-1\}$ *is a $\mathbb{Z}$-basis of $\mathcal{R} = \mathbb{Z}[x]/(\Phi_m(x))$.*
2. *The dual basis of $\mathbf{B}$ is*

$$(\mathbf{b}_1^\vee, \ldots, \mathbf{b}_n^\vee) = \frac{1 - x^{m'}}{m} \cdot (1, x, \ldots, x^{n-1}) \cdot (L_p \otimes I_{m'}), \tag{2.3}$$

where $L_p \in \mathbb{Z}^{(p-1)\times(p-1)}$ is a lower-triangular matrix with all entries 1 in its lower-left triangle and $I_{m'}$ is the $m' \times m'$ identity matrix. In other words, matrix $L_p \otimes I_{m'}$ is obtained from L_p by replacing every entry of value 1 with $I_{m'}$.

Tensor Field and Tensor Ring. Let K, L be two field extensions of $\mathbb{Q}$. Then the field tensor product $K \otimes L$ is the set of all $\mathbb{Q}$-linear combinations of *pure tensors* $a \otimes b$ for $a \in K, b \in L$, equipped with the following multiplication: for all $a_1, a_2 \in K$, $b_1, b_2 \in L$,

$$(a_1 \otimes b_1)(a_2 \otimes b_2) = (a_1 a_2) \otimes (b_1 b_2).$$

The tensor product of rings is defined in the same way, except that it is made up of only $\mathbb{Z}$-linear combinations of pure tensors.

An important fact is that the m-th cyclotomic number field $K = \mathbb{Q}(\xi_m) \cong \mathbb{Q}[x]/(\Phi_m(x))$ may be viewed as (i.e., is isomorphic to) the tensor product of prime-power cyclotomics:

$$K \cong \bigotimes_{i=1}^{\ell} K_i = \mathbb{Q}(\xi_{m_1}, \xi_{m_2}, \ldots, \xi_{m_\ell}),$$

where $m = \prod_{i=1}^{\ell} m_i$ is the prime-power factorization of m and $K_i = \mathbb{Q}(\xi_{m_i})$. Equivalently, K may be viewed as the multivariate polynomial field

$$K \cong \mathbb{Q}[x_1, x_2, \ldots, x_\ell] / (\Phi_{m_1}(x_1), \Phi_{m_2}(x_2), \ldots, \Phi_{m_\ell}(x_\ell)),$$

where there is one indeterminant x_i and modulus $\Phi_{m_i}(x_i)$ for every prime-power divisor m_i. Similar decompositions hold for the ring of integers $\mathcal{R} \cong \mathbb{Z}[x]/(\Phi_m(x))$ and the dual $\mathcal{R}^\vee$.

2.2 Homomorphic Encryption

Ring Learning with Errors. For a real number $\sigma > 0$, $\mathcal{DG}(\sigma^2)$ samples a vector in $\mathbb{Z}^n$ by dawning its coefficient independently form the discrete Gaussian distribution of variance σ^2. For an positive integer h, $\mathcal{HWT}(h)$ is the set of signed binary vectors in $\{0, \pm 1\}^n$ whose Hamming weight is exactly h.

Let χ be a distribution over $\mathcal{R}$ and $\sigma > 0$ be a real. Let $n = \deg(\Phi_m(x))$. The ring learning with errors (RLWE) assumption with respect to the parameters (n, Q, χ, σ) is the following: given polynomially many samples of either $(\mathbf{a}, \mathbf{b})$ or $(\mathbf{a}, \mathbf{a}\mathbf{s}_0 + \mathbf{e})$, where $\mathbf{a}, \mathbf{b} \leftarrow \mathcal{R}_Q$, $\mathbf{s}_0 \leftarrow \chi$, $e \leftarrow \mathcal{DG}(\sigma^2)$, it is computationally hard to distinguish between the two groups of samples. The most popular HE schemes such as FV [8], BGV [1] and CKKS [3] rely on the security provided by the RLWE assumption. In practical, χ is chosen to be $\mathcal{HWT}(h)$ with some positive integer h.

BGV Scheme [1]. The plaintext space and ciphertext space are $\mathcal{R}_p$ and $\mathcal{R}_{Q_i}$ respectively.

– $\underline{\mathsf{BGV.Setup}(1^\lambda, p, m, S)}$: For given security parameter λ, the plaintext prime modulus p, the order of primitive root of unity m, and a set $S \subset \mathbb{Z}_m^*$, choose a series of ciphertext moduli $\{Q_i, i = 1, \ldots, \ell\}$, a modulus P, a key distribution $\mathcal{HWT}(h)$, and an error parameter σ. The public parameters are $\mathsf{pp} = (m, p, Q_i, P, \chi, \sigma, S)$, $i = 1, \ldots, \ell$.

- $\underline{\mathsf{BGV.KeyGen}(\mathsf{pp})}$: Given public parameters pp, generate a secret key $\mathbf{s} = (-\mathbf{s_0}, 1)$ with $\mathbf{s_0} \leftarrow \mathcal{HWT}(h)$, a public key pk, a relinearization key rlk, and automorphism (key-switching) keys $\{\mathsf{atk}_i : i \in S\}$ of the automorphisms $\mathbf{m}(x) \mapsto \mathbf{m}(x^i)$ for all $i \in S$.
- $\underline{\mathsf{BGV.Enc}(\mathsf{pk}, \mathbf{m})}$: Given a public key pk and a plaintext $\mathbf{m} \in \mathcal{R}_p$, output a ciphertext $\mathbf{c}$ encrypting the plaintext $\mathbf{m}$.
- $\underline{\mathsf{BGV.Dec}(\mathbf{s}, \mathbf{c})}$: Given a ciphertext $\mathbf{c}$ with secret key $\mathbf{s}$, output the encrypted plaintext $\mathbf{m} \in \mathcal{R}_p$.
- $\underline{\mathsf{BGV.Add}(\mathbf{c}, \mathbf{c}')}$: Given two ciphertexts $\mathbf{c}$ and $\mathbf{c}'$ encrypting the plaintext $\mathbf{m}_1$ and $\mathbf{m}_2$ respectively, output a ciphertext $\mathbf{c}_{\text{add}}$ encrypting the plaintext $\mathbf{m}_1 + \mathbf{m}_2$.
- $\underline{\mathsf{BGV.Mult}(\mathsf{rlk}, \mathbf{c}, \mathbf{c}')}$: Given two ciphertexts $\mathbf{c}$ and $\mathbf{c}'$ encrypting the plaintext $\mathbf{m}_1$ and $\mathbf{m}_2$ respectively and a relinearization key rlk, output a ciphertext $\mathbf{c}_{\text{mul}}$ encrypting the plaintext $\mathbf{m}_1\mathbf{m}_2$.
- $\underline{\mathsf{BGV.CMult}(\mathbf{c}, \mathbf{f})}$: Given a ciphertexts $\mathbf{c}$ encrypting the plaintext $\mathbf{m}$, and given a plaintext $\mathbf{f} \in \mathcal{R}_p$, output a ciphertext $\mathbf{c}_{\text{cmul}}$ encrypting the plaintext $\mathbf{f} \cdot \mathbf{m}$.
- $\underline{\mathsf{BGV.Auto}(\mathsf{atk}_i, \mathbf{c})}$: Given a ciphertext $\mathbf{c}$ encrypting the plaintext $\mathbf{m}(x)$ and an automorphism key atk_i for some $i \in S$, output a ciphertext $\mathbf{c}_{\text{aut}}$ encrypting the plaintext $\mathbf{m}(x^i)$ with secret key $\mathbf{s}$.
- $\underline{\mathsf{BGV.ModulusSwitch}(\mathbf{c}, Q_i, Q_j)}$: Given a ciphertext $\mathbf{c} \in \mathcal{R}_{Q_i}^2$ encrypting the plaintext $\mathbf{m}(x)$ and two moduli Q_i and Q_j, with $Q_i > Q_j$, output a ciphertext $\mathbf{c}_{\text{switch}} \in \mathcal{R}_{Q_j}^2$ encrypting the plaintext $\mathbf{m}(x)$.

For convenience, we use $\mathsf{Enc}_{\mathbf{s}}(\mathbf{m})$ to denote the set of all BGV ciphertexts encrypting the plaintext $\mathbf{m}$ under secret key $\mathbf{s}$.

Canonical Embedding. Let $\xi := e^{2\pi\mathbf{i}/m} \in \mathbb{C}$ be a primitive m-th root of unity. Consider two polynomials $\mathbf{f}(x), \mathbf{g}(x) \in \mathbb{Z}[x]$ such that $\mathbf{f}(x) \equiv \mathbf{g}(x) \mod \Phi_m(x)$. For any primitive m-th root of unity ξ^j, where $j \in \mathbb{Z}_m^*$, since ξ^j is a root of $\Phi_m(x)$, $\mathbf{f}(\xi^j) = \mathbf{g}(\xi^j)$. This means that if $\mathbf{a} \in \mathcal{R}$ such that $\mathbf{a} \equiv \mathbf{f} \mod \Phi_m(x)$, then $\mathbf{a}(\xi^j) := \mathbf{f}(\xi^j)$ is well defined.

The canonical embedding of $\mathbf{a} \in \mathcal{R}$ is the vector obtained by evaluating $\mathbf{a}$ at all primitive m-th roots of unity:

$$\mathrm{can}(\mathbf{a}) := \left(\mathbf{a}\left(\xi^j\right)\right)_{j \in \mathbb{Z}_m^*} \in \mathbb{C}^{\varphi(m)}.$$

Define $\|\mathbf{a}\|_{\text{can}} := \|\mathrm{can}(\mathbf{a})\|_\infty = \max_{j \in \mathbb{Z}_m^*} |\mathbf{a}(\xi^j)|$. We say $\mathbf{a}$ is bound by B if $\|\mathbf{a}\|_{\text{can}} \leq B$.

Noise Estimate in BGV Scheme. For ciphertext $\mathbf{c} \in \mathcal{R}_Q^2$, the noise of $\mathbf{c}$ is defined as $\mathbf{e} = \langle \mathbf{c}, \mathbf{s} \rangle \mod Q$, where $\mathbf{s} \in \mathcal{R}_Q^2$ is the secret key. We say the noise of $\mathbf{c}$ is bounded by E if $\|\mathbf{e}\|_{\text{can}} \leq E$. We will recall the estimate of noise in the BGV scheme [11,13].

Lemma 2.2 (Modulus switch error estimate [6]). *Let $\mathbf{c} \in \mathcal{R}_Q^2$ be a BGV ciphertext encrypting $\mathbf{m} \in \mathcal{R}_p$ with noise bounded by E. Let q be another ciphertext modulus. Then* $\mathsf{BGV.ModulusSwitch}(\mathbf{c}, Q, q)$ *outputs a ciphertext in $\mathcal{R}_q^2$ encrypting $\mathbf{m} \in \mathcal{R}_p$ with noise bounded by $E + B_{\text{scale}}$ where $B_{\text{scale}} = p\big(\sqrt{3n} + 8\sqrt{nh/3}\big)$ is the bound of rounding error.*

Lemma 2.3 (Homomorphic automorphism error estimate [6]). *Let $\mathbf{c} \in \mathcal{R}_Q^2$ be a BGV ciphertext encrypting $\mathbf{m} \in \mathcal{R}_p$ with noise bounded by E. Let i be an integer in $\mathbb{Z}^*$ and the gadget vector $\boldsymbol{g}$ used in key-switching be $(1, 2, \ldots, 2^{\lfloor \log Q \rfloor})$. Then* $\mathsf{BGV.Auto}(\mathsf{atk}_i, \mathbf{c})$ *outputs a ciphertext $\mathbf{c}'$ encrypting $\mathbf{m}(x^i) \in \mathcal{R}_p$ with secret key $\mathbf{s}$, and noise of $\mathbf{c}'$ is bounded by $E + B_{\text{ks}}$, where $B_{\text{ks}} = \frac{16}{\sqrt{3}} pn\sigma \lceil \log Q \rceil$ is the bound of key-switching error, and σ is the standard variance of discrete Gaussian distribution used in sampling in the key-switching keys.*

Lemma 2.4 (Plaintext-ciphertext multiplication error estimate [11]). *Let $\mathbf{c} \in \mathcal{R}_Q^2$ be a BGV ciphertext encrypting $\mathbf{m} \in \mathcal{R}_p$ with noise bounded by E. Let $\mathbf{f} \in \mathcal{R}_p$ be a plaintext. Then* $\mathsf{BGV.CMult}(\mathbf{c}, \mathbf{f})$ *outputs a ciphertext in $\mathcal{R}_Q^2$ encrypting $\mathbf{f} \cdot \mathbf{m}$ with noise bounded by $E \cdot \|\mathbf{f}\|_{\text{can}}$.*

Lemma 2.5 (Ciphertext-ciphertext multiplication error estimate [13]). *Let $\mathbf{c}_l \in \mathcal{R}_{Q_l}^2$ be two BGV ciphertexts with moduli Q_l encrypting $\mathbf{m}_l$ with noise bounded by E_l respectively, for $l = 1, 2$. Let q be a ciphertext modulus satisfying $q \mid Q_l$, for $l = 1, 2$, and*

$$q \approx \min\big\{(B_{\text{scale}} \cdot Q_1)/E_1, (B_{\text{scale}} \cdot Q_2)/E_2\big\}. \tag{2.4}$$

Then $\mathsf{BGV.Mult}(\mathsf{rlk}, \mathbf{c}_1, \mathbf{c}_2)$ *outputs a ciphertext in $\mathcal{R}_q^2$ encrypting $\mathbf{m}_1 \cdot \mathbf{m}_2$ with noise bounded by $\Big((q/Q_1)E_1 + B_{\text{scale}}\Big)\Big((q/Q_2)E_2 + B_{\text{scale}}\Big) + B_{\text{ks}}$.*

We want to emphasize that, to choose a proper common ciphertext modulus q, in HElib [13], the modulus switching is performed before ciphertext multiplication instead of after ciphertext multiplication, which is different from the original BGV scheme.

3 Plaintext Matrix Multiplication via Tensor Ring

3.1 Notations.

Recall the cyclotomic field $K := \mathbb{Q}[x]/(\Phi_m(x))$. In our new framework, we choose $m = m_1 m_2 m_3$ with m_1, m_2 and m_3 pairwise coprime. In particular, we choose $m_1 = p_1^{a_1}$, $m_2 = p_2^{a_2}$ and $m_3 = p_3^{a_3}$, where p_1, p_2 and p_3 are primes, such that $\varphi(m_1) \approx \varphi(m_2) \approx \varphi(m_3)$ asymptotically. In Section 2.1, we mentioned that $K \cong \mathbb{Q}[x, y, z]/(\phi_{m_1}(x), \phi_{m_2}(y), \phi_{m_3}(z))$. Therefore, we can view K as $\mathbb{Q}[x, y, z]/(\phi_{m_1}(x), \phi_{m_2}(y), \phi_{m_3}(z))$. Let $K_i = \mathbb{Q}[x]/(\phi_{m_i}(x))$ for $i = 1, 2, 3$. Then $K \cong K_1 \otimes K_2 \otimes K_3$. Let $\mathcal{R} = \mathbb{Z}[x, y, z]/(\phi_{m_1}(x), \phi_{m_2}(y), \phi_{m_3}(z))$. Let $\mathcal{R}_i = \mathbb{Z}[x]/(\phi_{m_1}(x))$, for $i = 1, 2, 3$. Then $\mathcal{R} \cong \mathcal{R}_1 \otimes \mathcal{R}_2 \otimes \mathcal{R}_3$. The following notations are introduced in [19]:

- K_{12}, K_{13} and K_{23} denote $K_1 \otimes K_2$, $K_1 \otimes K_3$ and $K_2 \otimes K_3$, respectively.
- $\mathcal{R}_{12}$, $\mathcal{R}_{13}$ and $\mathcal{R}_{23}$ denote $\mathcal{R}_1 \otimes \mathcal{R}_2$, $\mathcal{R}_1 \otimes \mathcal{R}_3$ and $\mathcal{R}_2 \otimes \mathcal{R}_3$, respectively.
- Set $r := \varphi(m_1)$, $s := \varphi(m_2)$ and $t := \varphi(m_3)$.
- $\{\mathbf{u}_1, \dots, \mathbf{u}_r\}$, $\{\mathbf{v}_1, \dots, \mathbf{v}_s\}$ and $\{\mathbf{w}_1, \dots, \mathbf{w}_t\}$ denote the $\mathbb{Z}$-bases of $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$, respectively.
- $\{\mathbf{u}_1^\vee, \dots, \mathbf{u}_r^\vee\}$, $\{\mathbf{v}_1^\vee, \dots, \mathbf{v}_s^\vee\}$ and $\{\mathbf{w}_1^\vee, \dots, \mathbf{w}_t^\vee\}$ denote the corresponding $\mathbb{Z}$-bases of the dual lattices $\mathcal{R}_1^\vee$, $\mathcal{R}_2^\vee$ and $\mathcal{R}_3^\vee$, respectively.

More notations and technologies will be used in this paper:

- Let $d_0 = \min\{r, s, t\}$ and $n = r \cdot s \cdot t$. Then $d_0 \approx n^{\frac{1}{3}}$.
- $\mathbb{Z}^{d \times d}$ denote the set of all integer matrices of size $d \times d$. A matrix said to be *compact sized* if $d \leq d_0$, and *large sized* if $d > d_0$.
- For $A = (a_{ij}) \in \mathbb{Z}^{d_0 \times d_0}$, let

$$\mathbf{m}_A^{(\mathbf{uv})} = \sum_{i,j}^{d} a_{ij} \mathbf{u}_i \mathbf{v}_j, \mathbf{m}_A^{(\mathbf{uw})} = \sum_{i,j}^{d} a_{ij} \mathbf{u}_i \mathbf{w}_j, \text{ and } \mathbf{m}_A^{(\mathbf{w}^\vee \mathbf{v})} = \sum_{i,j}^{d} a_{ij} \mathbf{w}_i^\vee \mathbf{v}_j. \tag{3.1}$$

- For any polynomial $\mathbf{a} \in \mathcal{R}$, the saying $\mathbf{a} \in \mathcal{R}_p$ means taking $\mathbf{a}$ as $\mathbf{a} \mod p$.

3.2 Multiplication of Compact Sized Matrix via Polynomial Operations

For input matrices $A, B \in \mathbb{Z}^{d \times d}$, to compute AB, we encode them respectively as:

$$\begin{aligned} A = (a_{ij})_{i,j=1,\dots,d} \in \mathbb{Z}^{d \times d} &\xrightarrow{\text{Encode}} \mathbf{m}_A^{(\mathbf{uw})} = \sum_{i=1}^{d} \sum_{j=1}^{d} a_{ij} \mathbf{u}_i \mathbf{w}_j, \\ B = (b_{ij})_{i,j=1,\dots,d} \in \mathbb{Z}^{d \times d} &\xrightarrow{\text{Encode}} \mathbf{m}_B^{(\mathbf{w}^\vee \mathbf{v})} = \sum_{k=1}^{d} \sum_{l=1}^{d} b_{kl} \mathbf{w}_k^\vee \mathbf{v}_l. \end{aligned} \tag{3.2}$$

We claim that the matrix product $A \cdot B$ can be computed using polynomial multiplications in the field K and then a trace operation.

Theorem 3.2. *Let $A = (a_{ij})$ and $B = (b_{ij})$ be two $d \times d$ matrices. Then*

$$\mathrm{Tr}_{K/K_{12}}(\mathbf{m}_A^{(\mathbf{uw})} \cdot \mathbf{m}_B^{(\mathbf{w}^\vee \mathbf{v})}) = \sum_{i=1}^{d} \sum_{l=1}^{d} c_{il} \mathbf{u}_i \mathbf{v}_l, \text{ with } (c_{ij})_{1 \leq i,j \leq d} = A \cdot B. \tag{3.3}$$

Proof. For all $i, j = 1, \dots, d$, since $\mathbf{u}_i$ and $\mathbf{v}_j$ both belong to K_{12}, $\sigma(\mathbf{u}_i) = \mathbf{u}_i$ and $\sigma(\mathbf{v}_j) = \mathbf{v}_j$ for any $\sigma \in \mathrm{Gal}(K/K_{12})$. So

$$\begin{aligned} \mathrm{Tr}_{K/K_{12}} \left(\mathbf{m}_A^{(\mathbf{uw})} \cdot \cdot \mathbf{m}_B^{(\mathbf{w}^\vee \mathbf{v})} \right) &= \mathrm{Tr}_{K/K_{12}} \left(\sum_{i,j,k,l} a_{ij} b_{kl} \mathbf{u}_i \mathbf{w}_j \mathbf{w}_k^\vee \mathbf{v}_l \right) \\ &= \sum_{i,j,k,l} a_{ij} b_{kl} \mathbf{u}_i \left(\sum_{\sigma \in \mathrm{Gal}(K/K_{12})} \sigma(\mathbf{w}_j \mathbf{w}_k^\vee) \right) \mathbf{v}_l \end{aligned} \tag{3.4}$$

Notice that $\mathrm{Gal}(K/K_{12})$ is isomorphic to $\mathrm{Gal}(K_3/\mathbb{Q})$ by the restriction map $\sigma \mapsto \sigma|_{K_3}$ for all $\sigma \in \mathrm{Gal}(K/K_{12})$. Since $\mathbf{w}_j\mathbf{w}_k^{\vee} \in K_3$, we have

$$\begin{aligned}\sum_{\sigma\in\mathrm{Gal}(K/K_{12})} \sigma(\mathbf{w}_j\mathbf{w}_k^{\vee}) &= \sum_{\sigma\in\mathrm{Gal}(K_3/\mathbb{Q})} \sigma(\mathbf{w}_j\mathbf{w}_k^{\vee}) \\ &= \mathrm{Tr}_{K_3/\mathbb{Q}}(\mathbf{w}_j\mathbf{w}_k^{\vee}) = \begin{cases} 1, & j = k, \\ 0, & j \neq k. \end{cases}\end{aligned} \tag{3.5}$$

Substituting it into (3.4), we get

$$\mathrm{Tr}_{K/K_{12}}\left(\mathbf{m}_A^{(\mathbf{uw})} \cdot \mathbf{m}_B^{(\mathbf{w}^{\vee}\mathbf{v})}\right) = \sum_{i,l=1}^{d}\left(\sum_{j=1}^{d} a_{ij}b_{jl}\right)\mathbf{u}_i\mathbf{v}_l = \sum_{i,l=1}^{d} c_{il}\mathbf{u}_i\mathbf{v}_l. \tag{3.6}$$

namely, $(c_{il})_{1\le i,l\le d} = A \cdot B$. □

3.3 Basis Switching

In Theorem 3.2, for two matrices encoded by the $\mathbf{u}_i\mathbf{w}_j$ and the $\mathbf{w}_i^{\vee}\mathbf{v}_j$ respectively, $C = AB$ is encoded by the $\mathbf{u}_i\mathbf{v}_j$. However, for homomorphic addition and iterative multiplication, it is preferable that the input and output are encoded in the same manner, for example, both A and B are encoded by the $\mathbf{u}_i\mathbf{v}_j$. In this situation, basis switch is necessary. Let A and B be both encoded by $\{\mathbf{u}_i\mathbf{v}_j : 1 \le i, j \le d\}$, i.e.,

$$\mathbf{m}_A^{(\mathbf{uv})} = \sum_{i=1}^{d}\sum_{j=1}^{d} a_{ij}\mathbf{u}_i\mathbf{v}_j, \quad \mathbf{m}_B^{(\mathbf{uv})} = \sum_{i=1}^{d}\sum_{j=1}^{d} b_{ij}\mathbf{u}_i\mathbf{v}_j. \tag{3.7}$$

Theorem 3.3. *Let* $\mathbf{f} = \sum_{j=1}^{d} \mathbf{v}_j^{\vee}\mathbf{w}_j \in K_{23}$ *and* $\mathbf{g} = \sum_{i=1}^{d} \mathbf{w}_i^{\vee}\mathbf{u}_i^{\vee} \in K_{23}$. *Then*

$$\begin{aligned}\mathrm{Tr}_{K/K_{13}}(\mathbf{m}_A^{(\mathbf{uv})} \cdot \mathbf{f}) &= \sum_{i,j} a_{ij}\mathbf{u}_i\mathbf{w}_j := \mathbf{m}_A^{(\mathbf{uw})}, \\ \mathrm{Tr}_{K/K_{23}}(\mathbf{m}_B^{(\mathbf{uv})} \cdot \mathbf{g}) &= \sum_{i,j} b_{ij}\mathbf{w}_i^{\vee}\mathbf{v}_j := \mathbf{m}_B^{(\mathbf{w}^{\vee}\mathbf{v})}.\end{aligned}$$

Proof. The proof is the same as in Theorem 3.2.

3.4 Trace Computation

Let K/F be a Galois extension and let $r = |\mathrm{Gal}(K/F)|$. Computing the trace directly by definition requires $r - 1$ automorphisms. On the other hand, Galois group $\mathrm{Gal}(\mathbb{Q}(\xi_m)/\mathbb{Q})$ is isomorphic to the multiplicative group $\mathbb{Z}_m^*$, which is cyclic if and only if n is 1, 2, 4, p^k or $2p^k$, where p is an odd prime and $k > 0$ [26]. Based on these properties, we apply a technique [12] to compute the trace function, which requires only $O(\log r)$ automorphisms.

Proposition 3.1. *Let K/F be a Galois extension. Suppose* $\mathrm{Gal}(K/F)$ *is cyclic. Let σ be a generator of* $\mathrm{Gal}(K/F)$*, then for any $\mathbf{m} \in K$ and $1 \leq \ell \leq r = |\mathrm{Gal}(K/F)|$, $\sum_{i=0}^{\ell-1} \sigma^i(\mathbf{m})$ can be calculated using at most $2\log \ell$ additions and $2\log \ell$ automorphisms. In particular,* $\mathrm{Tr}_{K/F}$ *can be calculated using at most $2\log r$ additions and $2\log r$ automorphisms.*

Proof. For $\ell = 1$, the statement holds trivial. Assume that the statement holds for $\ell \leq k-1$. For $\ell = k$, if ℓ is even, then

$$\sum_{i=0}^{\ell-1} \sigma^i(\mathbf{m}) = \sum_{i=0}^{\ell/2-1} \sigma^i(\mathbf{m}) + \sigma^{\ell/2}\left(\sum_{i=0}^{\ell/2-1} \sigma^i(\mathbf{m})\right). \tag{3.8}$$

According to the inductive hypothesis, $\sum_{i=0}^{\ell/2-1} \sigma^i(\mathbf{m})$ can be calculated using at most $2\log(\ell/2)$ additions and $2\log(\ell/2)$ automorphisms. By (3.8), $\sum_{i=0}^{\ell-1} \sigma^i(\mathbf{m})$ can be computed using at most $2\log(\ell/2)+1 < 2\log \ell$ additions and $2\log(\ell/2)+1 < 2\log \ell$ automorphisms.

If ℓ is odd, then

$$\sum_{i=0}^{\ell-1} \sigma^i(\mathbf{m}) = \sum_{i=0}^{(\ell-1)/2-1} \sigma^i(\mathbf{m}) + \sigma^{(\ell-1)/2}\left(\sum_{i=0}^{(\ell-1)/2-1} \sigma^i(\mathbf{m})\right) + \sigma^{\ell-1}(\mathbf{m}). \tag{3.9}$$

According to the inductive hypothesis, $\sum_{i=0}^{(\ell-1)/2-1} \sigma^i(\mathbf{m})$ can be calculated using at most $2\log((\ell-1)/2)$ additions and $2\log((\ell-1)/2)$ automorphisms. By (3.9), $\sum_{i=0}^{\ell-1} \sigma^i(\mathbf{m})$ can be computed using at most $2\log((\ell-1)/2)+2 = 2\log(\ell-1) < 2\log \ell$ additions and $2\log((\ell-1)/2)+2 = 2\log(\ell-1) < 2\log \ell$ automorphisms. □

Based on the proof of Proposition 3.1, we get Algorithm 1 for trace computation.

As $\mathrm{Gal}(K/K_{12})$, $\mathrm{Gal}(K/K_{13})$ and $\mathrm{Gal}(K/K_{23})$ are isomorphic to $\mathrm{Gal}(K_3/\mathbb{Q})$, $\mathrm{Gal}(K_2/\mathbb{Q})$ and $\mathrm{Gal}(K_1/\mathbb{Q})$ respectively, if for $i = 1, 2, 3$, $m_i = p_i^{s_i}$, where each p_i is an odd prime, then Proposition 3.1 guarantees that $\mathrm{Tr}_{K_i/\mathbb{Q}}$ can be computed at with most $2\log\varphi(m_i)$ additions and $2\log\varphi(m_i)$ automorphisms. In particular, if $m_i = 2^{s_i}$ for some i, then $\varphi(m_i) = 2^{s_i-1}$. Indeed, $\mathrm{Tr}_{K_i/\mathbb{Q}}$ can be computed using only $s_i - 1$ additions and $s_i - 1$ automorphisms by the tower structure of field extensions, see [19], Section 4.3.

3.5 Multiplication of Compact Sized Matrices

Our new framework for compact sized matrix multiplication consists of three subprograms: (1) encoding, which encodes a matrix by the basis $\mathbf{u}_i\mathbf{v}_j$, (2) decoding, which decodes the polynomial encoding a matrix by the basis $\mathbf{u}_i\mathbf{v}_j$, (3) matrix multiplication (Algorithm 2 below).

Algorithm 1: $\mathsf{Tr}_{K/F}(\mathbf{m}, r)$

```
Input :
  1. Polynomial $\mathbf{m} \in \mathcal{R}$.
  2. Integer $r$.
Output: A polynomial $\sum_{i=0}^{r-1} \sigma^i(\mathbf{m})$.
begin
    if $r = 1$ then
        return $\mathbf{m}$;
    end
    if $r$ is even then
        $\mathbf{m}_{r/2} = \mathsf{Tr}_{K/F}(\mathbf{m}, r/2)$;
        return $\mathbf{m}_{r/2} + \sigma^{r/2}(\mathbf{m}_{r/2})$;
    else
        $\mathbf{m}_{(r-1)/2} = \mathsf{Tr}(\mathbf{m}, (r-1)/2)$;
        return $\mathbf{m}_{(r-1)/2} + \sigma^{(r-1)/2}(\mathbf{m}_{(r-1)/2}) + \sigma^{r-1}(\mathbf{m})$;
    end
end
```

Algorithm 2: Compact sized matrix multiplication

```
Input : $\mathbf{m}_A^{(\mathbf{uv})}, \mathbf{m}_B^{(\mathbf{uv})} \in \mathcal{R}$ that encode two $d \times d$ matrices $A$ and $B$ by
        $\{\mathbf{u}_i \mathbf{v}_j : 1 \le i, j \le d\}$, respectively.
Output: A polynomial $\mathbf{m}_{AB}^{(\mathbf{uv})} \in \mathcal{R}$ encoding $A \cdot B$.
begin
    $\mathbf{m}_A^{(\mathbf{uw})} = \mathrm{Tr}_{K/K_{13}}(\mathbf{m}_A^{(\mathbf{uv})} \cdot \mathbf{f})$; // Basis Switching
    $\mathbf{m}_B^{(\mathbf{w}^\vee \mathbf{v})} = \mathrm{Tr}_{K/K_{23}}(\mathbf{m}_B^{(\mathbf{uv})} \cdot \mathbf{g})$; // Basis Switching
    $\mathbf{m}_{AB}^{(\mathbf{uv})} = \mathrm{Tr}_{K/K_{12}}(\mathbf{m}_A^{(\mathbf{uw})} \cdot \mathbf{m}_B^{(\mathbf{w}^\vee \mathbf{v})})$;
    return $\mathbf{m}$;
end
```

We analyze the number of polynomial multiplications and automorphisms in Algorithm 2. In lines 2, 3, and 4, each line require one polynomial multiplication, resulting in a total of three polynomial multiplications. The trace operations in lines 2, 3, and 4 each require at most $2 \log r$, $2 \log s$, and $2 \log t$ automorphisms, respectively. Therefore, the algorithm calls for at most $2(\log r + \log s + \log t) = 2 \log n$ automorphisms, where $n = rst$ is the dimension of $\mathcal{R}$. In summary, the algorithm requires three polynomial multiplications and $O(\log d)$ automorphisms.

3.6 Multiplication of Large Sized Matrices

If $d > d_0 = \min\{r, s, t\}$, we need to divide every input matrix into $L \times L$ blocks, where each block has size $d_0 \times d_0$, so $L = \lceil d/d_0 \rceil$. For two input $d \times d$ matrices

A, B, the decomposition gives

$$A = \begin{pmatrix} A^{(11)} & \cdots & A^{(1L)} \\ \vdots & \ddots & \vdots \\ A^{(L1)} & \cdots & A^{(LL)} \end{pmatrix}, \text{ with } A^{(kl)} = (a_{ij}^{(kl)}) \in \mathbb{Z}^{d_0 \times d_0}, 1 \le k, l \le L.$$
$$B = \begin{pmatrix} B^{(11)} & \cdots & B^{(1L)} \\ \vdots & \ddots & \vdots \\ B^{(L1)} & \cdots & B^{(LL)} \end{pmatrix}, \text{ with } B^{(kl)} = (b_{ij}^{(kl)}) \in \mathbb{Z}^{d_0 \times d_0}, 1 \le k, l \le L. \tag{3.10}$$

Then each block is encoded by $\{\mathbf{u}_i\mathbf{v}_j : 1 \le i, j \le d_0\}$, i.e,

$$\mathbf{m}_A^{(kl)} = \sum_{i=1}^{d_0}\sum_{j=1}^{d_0} a_{ij}^{(kl)}\mathbf{u}_i\mathbf{v}_j \quad \text{and} \quad \mathbf{m}_B^{(kl)} = \sum_{i=1}^{d_0}\sum_{j=1}^{d_0} b_{ij}^{(kl)}\mathbf{u}_i\mathbf{v}_j,$$

with $1 \le k, l \le L$.

Algorithm 3: Block Matrix Multiplication

```
Input  : $2L^2$ elements $\{\mathbf{m}_A^{(kl)} : 1 \le k, l \le L\}$ and $\{\mathbf{m}_B^{(kl)} : 1 \le k, l \le L\}$ of $\mathcal{R}$
         encoding two $d \times d$ matrices $A$ and $B$ by $\{\mathbf{u}_i\mathbf{v}_j : 1 \le i, j \le d_0\}$.
Output: $L^2$ elements $\{\mathbf{m}_{AB}^{(kl)} : 1 \le k, l \le L\}$ of $\mathcal{R}$ that encode $A \cdot B$.
1  begin
2      for $k, l = 1$ to $L$ do
3          $\tilde{\mathbf{m}}_A^{(kl)} = \mathrm{Tr}_{K/K_{13}}(\mathbf{m}_A^{(kl)} \cdot \mathbf{f})$ // Bases Switching;
4          $\tilde{\mathbf{m}}_B^{(kl)} = \mathrm{Tr}_{K/K_{23}}(\mathbf{m}_B^{(kl)} \cdot \mathbf{g})$; // Bases Switching
5      end
6      for $k, l = 1$ to $L$ do
7          $\tilde{\mathbf{m}}^{(kl)} = 0$;
8          for $h = 1$ to $L$ do
9              $\tilde{\mathbf{m}}^{(kl)} = \tilde{\mathbf{m}}^{(kl)} + \tilde{\mathbf{m}}_A^{(kh)} \cdot \tilde{\mathbf{m}}_B^{(hl)}$
10         end
11     end
12     for $k, l = 1$ to $L$ do
13         $\mathbf{m}_{AB}^{(kl)} = \mathrm{Tr}_{K/K_{12}}(\tilde{\mathbf{m}}^{(kl)})$;
14     end
15     return $\{\mathbf{m}_{AB}^{(kl)} : 1 \le k, l \le L\}$;
16 end
```

Theorem 3.4. *Let A and B be decomposed as in (3.10). Suppose that*

$$AB = C = \begin{pmatrix} C^{(11)} & \cdots & C^{(1L)} \\ \vdots & \ddots & \vdots \\ C^{(L1)} & \cdots & C^{(LL)} \end{pmatrix}, \text{ with } C^{(kl)} = (c_{ij}^{(kl)}) \in \mathbb{Z}^{d_0 \times d_0}, 1 \le k, l \le L.$$

Then Algorithm 3 returns a set $\{\mathbf{m}_{AB}^{(kl)} : 1 \le k, l \le L\}$ *with*

$$\mathbf{m}_{AB}^{(kl)} = \sum_{i=1}^{d_0} \sum_{j=1}^{d_0} c_{ij}^{(kl)} \mathbf{u}_i \mathbf{v}_j. \tag{3.11}$$

Proof. According to Theorem 3.3, in lines 3 and 4 of Algorithm 3, for all $k, l = 1, \ldots, L$,

$$\tilde{\mathbf{m}}_A^{(kl)} = \sum_{i=1}^{d_0} \sum_{j=1}^{d_0} a_{ij}^{(kl)} \mathbf{u}_i \mathbf{w}_j, \qquad \tilde{\mathbf{m}}_B^{(kl)} = \sum_{i=1}^{d_0} \sum_{j=1}^{d_0} b_{ij}^{(kl)} \mathbf{w}_i^{\vee} \mathbf{v}_j.$$

By the computations from line 6 to line 11, $\tilde{\mathbf{m}}^{(kl)} = \sum_{h=1}^{L} \tilde{\mathbf{m}}_A^{(kh)} \cdot \tilde{\mathbf{m}}_B^{(hl)}$, for $1 \le k, l \le L$. Therefore, in line 13, we have $\mathbf{m}_{AB}^{(kl)} = \sum_{h=1}^{L} \mathrm{Tr}_{K/K_{12}}(\tilde{\mathbf{m}}_A^{(kh)} \cdot \tilde{\mathbf{m}}_B^{(hl)})$. By Theorem 3.2, for all $1 \le h, k, l \le L$, by letting $(c_{ij}^{(klh)})_{1 \le i,j \le d_0} = A^{(kh)} B^{(hl)}$ and $(c_{ij}^{(kl)})_{1 \le i,j \le d_0} = \sum_{h=1}^{L} A^{(kh)} B^{(hl)}$, we have

$$\mathbf{m}_{AB}^{(kl)} = \sum_{h=1}^{L} \left(\sum_{i=1}^{d_0} \sum_{j=1}^{d_0} c_{ij}^{(klh)} \mathbf{u}_i \mathbf{v}_j \right) = \sum_{i=1}^{d_0} \sum_{j=1}^{d_0} c_{ij}^{(kl)} \mathbf{u}_i \mathbf{v}_j, \ .$$

□

Next, we analyze the number of polynomial multiplications and automorphisms in Algorithm 3. From lines 2 to 5, the algorithm requires a total of $2L^2$ polynomial multiplications and at most $2L^2(\log r + \log s)$ automorphisms. From lines 6 to 11, the algorithm requires L^3 polynomial multiplications. Using Strassen algorithm can reduce the polynomial multiplications from L^3 to $O(L^{\log_2 7})$. The number of automorphisms from lines 12 to 14 is at most $2L^2 \log t$. Therefore, the algorithm calls for at most $2L^2(\log r + \log s + \log t) = 2L^2 \log n$ automorphisms and $O(L^{\log_2 7})$ polynomial multiplications.

Further assume $d = n^{\rho}$. Since $d_0 \approx n^{\frac{1}{3}}$, $L = \lceil \frac{d}{d_0} \rceil = O(n^{\rho - \frac{1}{3}}) = O(d^{1 - \frac{1}{3\rho}})$. The algorithm requires a total of $O(L^{\log_2 7}) = O\big(d^{(1 - \frac{1}{3\rho}) \log_2 7}\big)$ polynomial multiplications and $2L^2 \log n = O(d^{2 - \frac{2}{3\rho}} \log d)$ automorphisms.

4 Homomorphic Matrix Multiplication via Tensor Ring

In this section, we assume that p and m are coprime, where plaintext modulus p is a prime number. For $i = 1, 2, 3$, suppose that $m_i = p_i^{a_i}$, where each p_i is an odd prime. Then $\mathrm{Gal}(K/K_{12})$, $\mathrm{Gal}(K/K_{23})$ and $\mathrm{Gal}(K/K_{13})$ are all cyclic.

4.1 Homomorphic Trace Computation

Algorithm 1 can be extended directly to the homomorphic case, see Algorithm 4 below, where σ is a generator of $\mathrm{Gal}(K/F)$, F is one of K_{12}, K_{13} or K_{23}, and $r = |\mathrm{Gal}(K/F)|$.

Algorithm 4: $\mathsf{HomTr}_{K/F}(\mathbf{c}, r)$

```
Input :
  1. A ciphertext $\mathbf{c} \in \mathcal{R}_Q^2$ encrypting $\mathbf{m} \in \mathcal{R}_p$.
  2. An integer $r$.
Output: A ciphertext $\mathbf{c}_r$ encrypting $\sum_{i=0}^{r-1} \sigma^i(\mathbf{m}) \in \mathcal{R}_p$.
1  begin
2      if $r = 1$ then
3          return $\mathbf{c}$;
4      end
5      if $r$ is even then
6          $\mathbf{c}_{r/2} = \mathsf{HomTr}_{K/F}(\mathbf{c}, r/2)$;
7          return $\mathbf{c}_{r/2} + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^{r/2}}, \mathbf{c}_{r/2})$;
8      else
9          $\mathbf{c}_{(r-1)/2} = \mathsf{HomTr}_{K/F}(\mathbf{c}, (r-1)/2)$;
10         return $\mathbf{c}_{(r-1)/2} + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^{\frac{r-1}{2}}}, \mathbf{c}_{(r-1)/2}) + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^r}, \mathbf{c})$;
11     end
12 end
```

Theorem 4.5. *Let $\mathbf{c} \in \mathcal{R}_Q^2$ be a BGV ciphertext encrypting $\mathbf{m} \in \mathcal{R}_p$ with noise $\mathbf{e}$ bounded by E. Then $\mathsf{HomTr}_{K/F}(r, \mathbf{c})$ outputs a ciphertext $\mathbf{c}_r$ encrypting $\mathrm{Tr}_{K/F}(\mathbf{m})$ with noise $\mathbf{e}_r$ bounded by $rE+(r-1)B_{\mathrm{ks}}$, where $B_{\mathrm{ks}} = \frac{16}{\sqrt{3}} pn\sigma\lceil \log Q \rceil$, and σ is the standard variance of discrete Gaussian distribution used in sampling in the key-switching keys.*

Proof. The correctness of Algorithm 4 is based on the correctness of Algorithm 1 and homomorphic automorphism. We next prove the conclusion on the noise noise bound of $\mathbf{c}_r$.

When $\ell = 1$, then the algorithm outputs $\mathbf{c}$ directly, the conclusion holds trivially. Assume that the conclusion holds for $\ell < k$. For $\ell = k$, if ℓ is even, then

$$\mathbf{c}_\ell = \mathbf{c}_{\ell/2} + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^{\ell/2}}, \mathbf{c}_{\ell/2}). \tag{4.1}$$

By inductive hypothesis, the noise $\mathbf{e}_{\ell/2}$ of $\mathbf{c}_{\ell/2}$ is bounded by $\frac{\ell}{2}(E + B_{\mathrm{ks}}) - B_{\mathrm{ks}}$. Then by (4.1), the noise $\mathbf{e}_\ell$ of $\mathbf{c}_l$ satisfies

$$\|\mathbf{e}_\ell\|_{\mathrm{can}} \le \|\mathbf{e}_{\ell/2}\|_{\mathrm{can}} + \|\mathbf{e}_{\ell/2} + \mathbf{e}_{\mathrm{ks}}\|_{\mathrm{can}} \le l(E + B_{\mathrm{ks}}) - B_{\mathrm{ks}} \tag{4.2}$$

where $\mathbf{e}_{\mathrm{ks}}$ is the noise introduced by key-switching, which is bounded by E_{ks}.

If ℓ is odd, then

$$\mathbf{c}_\ell = \mathbf{c}_{(\ell-1)/2} + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^{(\ell-1)/2}}, \mathbf{c}_{\ell/2}) + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^\ell}, \mathbf{c}) \tag{4.3}$$

By inductive hypothesis, the noise $\mathbf{e}_{(\ell-1)/2}$ of $\mathbf{c}_{(\ell-1)/2}$ is bounded by $\frac{\ell-1}{2}(E + B_{\mathrm{ks}}) - B_{\mathrm{ks}}$. Then by (4.3), the noise $\mathbf{e}_\ell$ of $\mathbf{c}_\ell$ satisfies

$$\|\mathbf{e}_\ell\|_{\mathrm{can}} \le \|\mathbf{e}_{\frac{\ell-1}{2}}\|_{\mathrm{can}} + \|\mathbf{e}_{\frac{\ell-1}{2}} + \mathbf{e}_{\mathrm{ks}}\|_{\mathrm{can}} + \|\mathbf{e} + \mathbf{e}_{\mathrm{ks}}\|_{\mathrm{can}} \le \ell(E + B_{\mathrm{ks}}) - B_{\mathrm{ks}} \tag{4.4}$$

Therefore, the conclusion holds. □

Since Algorithm 4 requires at most $2\log r$ homomorphic automorphisms by Proposition 3.1, the algorithm requires $O(\log r)$ key-switching keys.

By Algorithm 4, we can homomorphically compute the trace $\mathrm{Tr}_{K/K_{12}}$, $\mathrm{Tr}_{K/K_{13}}$ and $\mathrm{Tr}_{K/K_{23}}$. These functions are denoted by $\mathsf{HomTr}_{K/K_{12}}$, $\mathsf{HomTr}_{K/K_{13}}$ and $\mathsf{HomTr}_{K/K_{23}}$, respectively.

Remark 1. A homomorphic trace algorithm is also proposed in [19]. where they considered a more general case that the ciphertext belongs to K^2. In our algorithm, the ciphertext belongs to $\mathcal{R}_Q^2$, so we use a simple approach similar to that in [12].

Moreover, we use the cyclic property of the Galois group to compute the trace function instead of the tower structure presented in [19]. This is because if m_i is a prime number, then the tower structure method does not work.

Remark 2. In fact, if $m_1 = 2^l$, then $r = \varphi(m_1) = 2^{l-1}$ and $\mathrm{Gal}(K/K_{23}) \simeq \langle\sigma\rangle \times \langle\tau\rangle$, where $\sigma^{r/2} = 1$ and $\tau^2 = 1$. We can first invoke Algorithm 4 to compute $\mathbf{c}_0 = \mathsf{Enc}_{\mathbf{s}}(\sum_{i=0}^{r/2-1} \sigma(\mathbf{m}))$, then compute $\mathbf{c}_1 = \mathbf{c}_0 + \mathsf{BGV.Auto}(\mathsf{atk}_\tau, \mathbf{c}_0)$ to obtain the trace. The number of homomorphic automorphisms is $\log r$. Since the noise of $\mathbf{c}_0$ is bounded by $(r/2)E + (r/2-1)B_{\mathrm{ks}}$ by Theorem 4.5, the noise of $\mathbf{c}_1$ is bounded by $rE + (r-1)B_{\mathrm{ks}}$.

4.2 Homomorphic Matrix Multiplication in the Compact Sized Case

In the case of $d \le d_0 = \min\{r, s, t\}$, since polynomial additions, polynomial multiplications and trace functions in Algorithm 2 and Algorithm 3 can be executed homomorphically, the framework proposed in Section 3 can extended to support homomorphic matrix multiplication. However, since $\mathbf{f}$ and $\mathbf{g}$ belong to $\mathcal{R}^\vee$ instead of $\mathcal{R}$, they have coefficients in $\mathbb{Q}$. So we need to make some modifications.

According to Lemma 2.1, $m_2 \cdot \mathbf{f} \in \mathcal{R}$ and $m_1 m_3 \cdot \mathbf{g} \in \mathcal{R}$. So, in the process of basis switching, we will multiply the ciphertexts by $m_2 \cdot \mathbf{f}$ and $m_1 m_3 \cdot \mathbf{g}$ instead of $\mathbf{f}$ and $\mathbf{g}$. This will cause the result of homomorphic matrix multiplication to be multiplied by m. This multiple needs to be recorded during the whole computation. As a result, a ciphertext encrypting a compact sized matrix is associated with an integer that record the exponent of the power of m.

We present a new framework for homomorphic matrix multiplication. It mainly consists of three subprograms: Encoding and Encryption (Algorithm 5), Decryption and Decoding (Algorithm 7), and Homomorphic Matrix Multiplication (Algorithm 6). In the following, we say a ciphertext $\mathbf{c} \in \mathcal{R}_Q^2$ encrypts a matrix A if $\mathbf{c}$ encrypts the plaintext $\mathbf{m}_A^{(\mathbf{uv})} \in \mathcal{R}_p$.

Noise Estimation. Below we first estimate the noise growth in lines 2 and 3 of Algorithm 6. By Lemma 2.4, we need to estimate the canonical norms of $\mathbf{f}$ and $\mathbf{g}$. First, we recall a lemma from [22,19].

Lemma 4.6 ([22,19]). *With notations introduced in Section 3.1, for all $i = 1, \ldots, d$,*

Algorithm 5: Encoding and encryption

```
Input  : A matrix $A = (a_{ij}) \in \mathbb{Z}_p^{d\times d}$ and the public key $\mathsf{pk}$;
Output: $(\mathbf{c}_A, 0)$, where $\mathbf{c}_A$ is a ciphertext encrypting $A$.
1 begin
2     $\mathbf{m}_A^{(\mathbf{uv})} = \sum_{i=1}^{d}\sum_{j=1}^{d} a_{ij}\mathbf{u}_i\mathbf{v}_j$;
3     $\mathbf{c}_A = \mathsf{BGV.Enc}(\mathsf{pk}, \mathbf{m}_A^{(\mathbf{uv})})$;
4     return $(\mathbf{c}_A,\, 0)$;
5 end
```

Algorithm 6: Homomorphic compact matrix multiplication

```
Input  :
   1. $(\mathbf{c}_{m^iA}, i)$ and $(\mathbf{c}_{m^jB}, j)$, where $\mathbf{c}_{m^jA}$ and $\mathbf{c}_{m^jB}$ are BGV ciphertexts encrypting
      two $d\times d$ matrices $m^iA, m^jB \in \mathbb{Z}_p^{d\times d}$ respectively;
   2. Automorphism keys for homomorphic traces computing;
   3. Relinearization key $\mathsf{rlk}$.
Output: $(\mathbf{c}, i+j+1)$, where $\mathbf{c}$ is a ciphertext encrypting $m^{i+j+1}AB \in \mathbb{Z}_p^{d\times d}$.
1 begin
2     $\mathbf{c}_1 = \mathsf{HomTr}_{K/K_{13}}(\mathsf{BGV.CMult}(\mathbf{c}_{m^iA}, m_2\cdot\mathbf{f}))$;
3     $\mathbf{c}_2 = \mathsf{HomTr}_{K/K_{23}}(\mathsf{BGV.CMult}(\mathbf{c}_{m^jB}, m_1m_3\cdot\mathbf{g}))$;
4     $\mathbf{c} = \mathsf{HomTr}_{K/K_{12}}(\mathsf{BGV.Mult}(\mathsf{rlk}, \mathbf{c}_1, \mathbf{c}_2))$;
5     return $(\mathbf{c}, i+j+1)$;
6 end
```

Algorithm 7: Decryption and decoding

```
Input  :
   1. $(\mathbf{c}_{m^kC}, k)$, where $\mathbf{c}_{m^kC}$ is a ciphertext encrypting a $d\times d$ matrix $m^kC$, where
      $C = (c_{ij}) \in \mathbb{Z}_p^{d\times d}$;
   2. secret key $\mathbf{s}$.
Output: Matrix $C \in \mathbb{Z}_p^{d\times d}$.
1 begin
2     $\mathbf{m}_{m^kC} = \mathsf{BGV.Dec}(\mathbf{s}, \mathbf{c}_{m^kC})$;
3     return $(c_{ij}\cdot m^{-k} \mod p)_{1\le i,j\le d}$; Suppose that $\mathbf{m}_{m^kC} = \sum_{i,j}^{d} c_{ij}\mathbf{u}_i\mathbf{v}_j$.
4 end
```

1. $\|\mathbf{u}_i\|_{\text{can}} = \|\mathbf{v}_i\|_{\text{can}} = \|\mathbf{w}_i\|_{\text{can}} = 1$;
2. $\|\mathbf{u}_i^\vee\|_{\text{can}} \le \frac{2(p_1-1)}{m_1}$, $\|\mathbf{v}_i^\vee\|_{\text{can}} \le \frac{2(p_2-1)}{m_2}$ *and* $\|\mathbf{w}_i^\vee\|_{\text{can}} \le \frac{2(p_3-1)}{m_3}$.

Corollary 4.1. *For the polynomials* $\mathbf{f}$ *and* $\mathbf{g}$ *defined in Theorem 3.3,*

$$\|m_2 \cdot \mathbf{f}\|_{\text{can}} \le 2n^{\frac{1}{3}}(p_2-1) \quad \text{and} \quad \|m_1m_3 \cdot \mathbf{g}\|_{\text{can}} \le 4n^{\frac{1}{3}}(p_1-1)(p_3-1).$$

Recall that the rounding noise is bounded by $B_{\text{scale}} = p\big(\sqrt{3n} + 8\sqrt{nh/3}\big)$ and noise introduced by key-switching is bounded by $B_{\text{ks}} = \frac{16}{\sqrt{3}}pn\sigma\lceil\log Q\rceil$ for $\mathbf{c} \in \mathcal{R}_Q^2$, where σ being the standard variance of discrete Gaussian distribution used in sampling in the key-switching keys [6].

Lemma 4.7. *Let* $\mathbf{c}_A, \mathbf{c}_B \in \mathcal{R}_Q^2$ *be two BGV ciphertexts encrypting* $\mathbf{m}_A^{(\mathbf{uv})}$ *and* $\mathbf{m}_B^{(\mathbf{uv})}$ *with noise* $\mathbf{e}_A$ *and* $\mathbf{e}_B$ *bounded by* E_1 *and* E_2*. Let*

$$\begin{aligned} \mathbf{c}_1 &= \mathsf{HomTr}_{K/K_{13}}(\mathsf{BGV.CMult}(\mathbf{c}_A, m_2 \cdot \mathbf{f})), \\ \mathbf{c}_2 &= \mathsf{HomTr}_{K/K_{23}}(\mathsf{BGV.CMult}(\mathbf{c}_B, m_1m_3 \cdot \mathbf{g})). \end{aligned} \tag{4.5}$$

Then $\mathbf{c}_1 \in \mathcal{R}_Q^2$ *encrypts* $m_2 \cdot \mathbf{m}_A^{(\mathbf{uw})}$ *with noise bounded by* E_1'*, and* $\mathbf{c}_2 \in \mathcal{R}_Q^2$ *encrypts* $m_1m_3 \cdot \mathbf{m}_B^{(\mathbf{w}^\vee\mathbf{v})}$ *with noise bounded by* E_2'*, where*

$$\begin{aligned} E_1' &= 2sn^{\frac{1}{3}}(p_2-1)E_1 + (s-1)B_{\text{ks}}, \\ E_2' &= 4rn^{\frac{1}{3}}(p_1-1)(p_3-1)E_2 + (r-1)B_{\text{ks}}. \end{aligned} \tag{4.6}$$

Proof. Since $\mathrm{Tr}_{K/K_{13}}(\mathbf{m}_A^{(\mathbf{uv})} \cdot m_2\mathbf{f}) = m_2 \cdot \mathbf{m}_A^{(\mathbf{uw})}$ by Theorem 3.3, $\mathbf{c}_1$ encrypts $m_2 \cdot \mathbf{m}_A^{(\mathbf{uw})} \in \mathcal{R}_p$ based on the correctness of Algorithm 4. By Corollary 4.1 and Lemma 2.4, the noise of $\mathsf{BGV.CMult}(\mathbf{c}_A, m_2 \cdot \mathbf{f})$ is bounded by $2n^{\frac{1}{3}}(p_2-1)E_1$. According to Theorem 4.5, the noise of $\mathbf{c}_1$ is bounded by $2sn^{\frac{1}{3}}(p_2-1)E_1 + (s-1)B_{\text{ks}}$. The statement about $\mathbf{c}_2$ can be proved similarly. □

Without loss of generality, suppose that the input ciphertexts of Algorithm 6 have the same modulus Q.

Theorem 4.6. *Let* $\mathbf{c}_{m^iA}, \mathbf{c}_{m^jB} \in \mathcal{R}_Q^2$ *be two BGV ciphertexts encrypting* $\mathbf{m}_{m^iA}^{(\mathbf{uv})}$ *and* $\mathbf{m}_{m^jB}^{(\mathbf{uv})}$ *with noise* $\mathbf{e}_1$ *and* $\mathbf{e}_2$*, respectively. Suppose that* $\|\mathbf{e}_1\|_{\text{can}} \le E_1$ *and* $\|\mathbf{e}_2\|_{\text{can}} \le E_2$*. Let* E_1'*,* E_2' *be defined in (4.6). Then Algorithm 6 outputs a ciphertext* $\mathbf{c} \in \mathcal{R}_q^2$ *encrypting* $\mathbf{m}_{m^{i+j+1}AB}^{(\mathbf{uv})}$ *with noise bounded by*

$$\begin{aligned} E &= t\left(\left(\frac{q}{Q}E_1' + B_{\text{scale}}\right)\left(\frac{q}{Q}E_2' + B_{\text{scale}}\right) + B_{\text{ks}}\right) + (t-1)B_{\text{ks}} \\ &= O(hp^2n^{\frac{4}{3}}), \end{aligned} \tag{4.7}$$

where $q \approx B_{\text{scale}} \cdot \min\{Q/E_1', Q/E_2'\}$*,* h *is the Hamming weight of the secret key, and* p *is the plaintext modulus.*

Proof. By Lemma 4.7, $\mathbf{c}_1$ in line 2 and $\mathbf{c}_2$ in line 3 of Algorithm 6 encrypt $\mathbf{m}^{(\mathbf{uw})}_{m_2 m^i A}$ and $\mathbf{m}^{(\mathbf{w}^\vee \mathbf{v})}_{m_1 m_3 m^j B}$ respectively with noise bounded by E'_1 and E'_2, respectively. By Theorem 3.2,

$$\mathrm{Tr}_{K/K_{12}}\left(\mathbf{m}^{(\mathbf{uw})}_{m_2 m^i A} \cdot \mathbf{m}^{(\mathbf{w}^\vee \mathbf{v})}_{m_1 m_3 m^j B}\right) = \mathbf{m}^{(\mathbf{uv})}_{m^{i+j+1} C}. \tag{4.8}$$

Then $\mathbf{c} \in \mathcal{R}_q^2$ in line 3 encrypts $\mathbf{m}^{(\mathbf{uv})}_{m^{i+j+1} C}$ by the correctness of Algorithm 4. By Lemma 2.5, the noise after ciphertext multiplication is bounded by

$$\left(\frac{q}{Q} E'_1 + B_{\text{scale}}\right)\left(\frac{q}{Q} E'_2 + B_{\text{scale}}\right) + B_{\text{sk}}.$$

where $(q/Q_\ell)E'_\ell$ is approximately equal to or less than B_{scale}, for $\ell = 1, 2$. Then by Theorem 4.5, after homomorphic trace operation, the noise of the ciphertext $\mathbf{c} \in \mathcal{R}_q^2$ is bounded by (4.7). According to the choice of q and $t = O(n^{\frac{1}{3}})$, $E = O(tB^2_{\text{scale}}) = O(hp^2 n^{\frac{4}{3}})$. □

Complexity Analysis. In lines 2 and 3 of Algorithm 6, each line requires one plaintext-ciphertext multiplication. Line 4 requires one ciphertext-ciphertext multiplication. The three homomorphic trace functions in lines 2, 3, and 4 require at most $2\log r$, $2\log s$, and $2\log t$ homomorphic automorphisms, respectively. Suppose that $d = n^\rho$. The algorithm calls for at most $2(\log r + \log s + \log t) = 2\log n = 2\log d^{\frac{1}{\rho}} = O(\log d)$ homomorphic automorphisms, two plaintext-ciphertext multiplications, and one plaintext-ciphertext multiplication.

4.3 Homomorphic Block Matrix Multiplication for Large Sized Matrices

The method for homomorphic block matrix multiplication can be easily derived by modifying Algorithm 3. Specifically, we need the following modifications to Algorithm 3:

- Each ciphertext is associated with an integer i to indicate that the ciphertext encrypts the m^i multiple of a matrix.
- Replace $\mathbf{f}$ and $\mathbf{g}$ in lines 3 and 4 with $m_2\mathbf{f}$ and $m_1 m_3 \mathbf{g}$, respectively.
- Replace the trace functions in lines 3,4 and 13 with the corresponding homomorphic trace functions.
- Replace the polynomial multiplications in lines 3 and 4 with plaintext-ciphertext multiplications.
- Replace the polynomial multiplication and addition in line 9 with ciphertext-ciphertext multiplication and addition.

The complexity can be derived from Section 3.6 directly. Suppose that $d = n^\rho$. The homomorphic block matrix multiplication requires $O(d^{2-\frac{2}{3\rho}})$ plaintext-ciphertext multiplications, $O(d^{(1-\frac{1}{3\rho})\log_2 7})$ ciphertext-ciphertext multiplications and $O(d^{2-\frac{2}{3\rho}} \log d)$ homomorphic automorphisms.

Packing and Unpacking. Recall that for large sized case, the input matrix is divided into $L \times L$ blocks, where each block is a $d_0 \times d_0$ matrix with $d_0 = \min\{r, s, t\}$. Encrypting each matrix directly will result in L^2 ciphertexts, which leads to a increase in all of the time encryption, the decryption time, and the communiation cost. In the following, we propose a technique to pack these L^2 matrices into one plaintext.

Suppose that $L^2 \leq t$. The matrix

$$A = \begin{pmatrix} A^{(1)} & \cdots & A^{(L)} \\ \vdots & \ddots & \vdots \\ A^{((L-1)L+1)} & \cdots & A^{(L^2)} \end{pmatrix} \in \mathbb{Z}_p^{Ld_0 \times Ld_0} \quad \text{with} \quad A^{(k)} = (a_{ij}^{(k)}),$$

is encoded by

$$\mathbf{m} = \sum_{k=1}^{L^2} \left(\sum_{i,j=1}^{d_0} M a_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j \right) m_3 \mathbf{w}_k^{\vee} \mod p, \tag{4.9}$$

where M is the inverse of m_3 in $\mathbb{Z}_p$. The client encrypts $\mathbf{m}$ and obtains $\mathbf{c} = \mathsf{Enc}_{\mathbf{s}}(\mathbf{m})$. When the server receives $\mathbf{c}$, it use Algorithm 8 below to obtain $\mathbf{c}_k = \mathsf{Enc}_{\mathbf{s}}(\mathbf{m}_k)$ with $\mathbf{m}_k = \sum_{i,j=1}^{d_0} a_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j$, for all $k = 1, \ldots, L^2$. After all homomorphic block matrix computation finishes, the server homomorphically packs the block matrices by Algorithm 9 and sends them to the client.

Algorithm 8: Homomorphic Blockwise Unpacking

```
Input  : A ciphertext $\mathbf{c}$ encrypting $\mathbf{m} = \sum_{k=1}^{L^2} M \cdot \mathbf{m}_k (m_3 \mathbf{w}_k^{\vee}) \in \mathcal{R}_p$
         with noise bounded by $B$, where $M$ is the inverse of $m_3$ in $\mathbb{Z}_p$
         and $\mathbf{m}_k = \sum_{i,j=1}^{d_0} a_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j$, for $k = 1, \ldots, L^2$;
Output: A ciphertexts $\{\mathbf{c}_k : k = 1, \ldots, L^2\}$ with each $\mathbf{c}_k$ encrypting
         $\mathbf{m}_k \in \mathcal{R}_p$.
1 begin
2     for $k = 1$ to $L^2$ do
3         $\mathbf{c}_k = \mathsf{HomTr}_{K/K_{12}}(\mathsf{BGV.CMult}(\mathbf{c}, \mathbf{w}_k))$
4     end
5     return $\{\mathbf{c}_k : k = 1, \ldots, L^2\}$;
6 end
```

Algorithm 9: Homomorphic Blockwise Packing

```
Input  : Ciphertexts $\{\mathbf{c}_k : k = 1, \ldots, L^2\}$, and each $\mathbf{c}_k$ encrypts
         $\mathbf{m}_k = \sum_{i,j=1}^{d_0} a_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j$;
Output: A ciphertext $\mathbf{c}$ encrypting $\mathbf{m} = \sum_{k=1}^{L^2} \left( \sum_{i,j=1}^{d_0} a_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j \right) \mathbf{w}_k$.
1 begin
2     $\mathbf{c} = \mathsf{BGV.CMult}(\mathbf{c}_1, \mathbf{w}_1)$;
3     for $k = 2$ to $L^2$ do
4         $\mathbf{c} = \mathsf{BGV.Add}(\mathbf{c}, \mathsf{BGV.CMult}(\mathbf{c}_k, \mathbf{w}_k))$
5     end
6     return $\mathbf{c}$;
7 end
```

Theorem 4.7 (Homomorphic blockwise unpacking). *Algorithm 8 outputs ciphertexts* $\{\mathbf{c}_k : k = 1, \ldots, L^2\}$, *where each* $\mathbf{c}_k$ *encrypts* $\mathbf{m}_k = \sum_{i,j=1}^{d_0} a_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j$ *with noise bounded by* $tB + (t-1)B_{\mathrm{ks}}$, *for all* $k = 1, \ldots, L^2$.

Proof. It is easy to verify that $\mathrm{Tr}_{K/K_{12}}(\mathbf{m} \cdot \mathbf{w}_k) \equiv \mathbf{m}_k \mod p$ for all $k = 1, \ldots, L^2$. Therefore, by Theorem 4.5, $\mathbf{c}_k$ encrypts $\mathbf{m}_k \in \mathcal{R}_p$ with noise bounded by $tB + (t-1)E_{\mathrm{ks}}$, for all $k = 1, \ldots, L^2$. □

Theorem 4.8 (Homomorphic blockwise packing). *Algorithm 8 outputs ciphertext* $\mathbf{c}$ *that encrypts* $\mathbf{m} = \sum_{k=1}^{L^2} \left(\sum_{i=1,j}^{d_0} a_{ij}^{(k)} \mathbf{u}_i \mathbf{v}_j \right) \mathbf{w}_k$ *with noise bounded by* $\sum_{i=1}^{L^2} E_i$.

Proof. The noise bound is deduced from $\|\mathbf{w}_i\|_{\mathrm{can}} = 1$. □

The maximal size of matrix that can be packed in a plaintext is $d \times d$, with $d = \lfloor t^{\frac{1}{2}} \rfloor \cdot d_0$. Since t and d_0 are both approximately equal to $n^{\frac{1}{3}}$, $d = \lfloor t^{\frac{1}{2}} \rfloor \cdot d_0 \approx n^{\frac{1}{3} \cdot \frac{1}{2}} \cdot n^{\frac{1}{3}} = O(n^{\frac{1}{2}})$.

4.4 Total Communication Cost

Using the pack-unpack method, it is possible to pack a matrix of size $d \times d$ in one plaintext if $d = O(n^{\frac{1}{2}})$. We need $O(\log r)$, $O(\log s)$ and $O(\log t)$ key-switching keys for the homomorphic trace functions $\mathsf{HomTr}_{K/K_{12}}$, $\mathsf{HomTr}_{K/K_{13}}$ and $\mathsf{HomTr}_{K/K_{23}}$ respectively. Therefore, a total of $O(\log n) = O(\log d)$ key-switching keys are needed.

In summary, when $d = O(n^{\frac{1}{2}})$, the communication cost is: one ciphertext per matrix, $O(\log d)$ key-switching keys, and one relinearization key.

5 Comparison with Previous Work

This section provides more detailed comparisons under specific parameters, the results are collected in Table 2.

Table 2. Comparison with Previous Work

d	Method	m	n	CMult	Aut	Mult	Ctxt[a]	Keys[b]
16	[20,27]	16384	8192	256	1024	256	16	4
	[4,23]	$17 \cdot 1024$	8192	16	80	16	1	19
	[17]	16384	8192	64	68	16	1	45
	[16]	$17 \cdot 1024$	8192	32	42	16	1	30
	[25]	16384	8192	32	42	1	1	42
	Ours	$17 \cdot 19 \cdot 25$	5440	2	14	1	1	14
32	[20,27]	16384	8192	1024	5120	1024	32	5
	[4,23]	$257 \cdot 128$	16384	32	191	32	1	36
	[17]	16384	8192	128	120	32	1	93
	[16]	$257 \cdot 128$	16384	64	82	32	1	60
	[25]	16384	8192	128	168	8	4	42
	Ours	$17 \cdot 19 \cdot 25$	5440	8	56	7	1	14
64	[20,27]	16384	8192	4096	24576	4096	64	6
	[4,23]	$257 \cdot 256$	32768	64	448	64	1	69
	[17]	16384	8192	256	225	64	1	189
	[16]	$257 \cdot 256$	32768	128	154	64	1	126
	[25]	16384	8192	512	672	49	16	42
	Ours	$17 \cdot 19 \cdot 25$	5440	32	224	49	1	14

[a] Ctxt: number of ciphertexts needed per matrix.
[b] Keys: number of key-switching keys.

Comparisons:

1. For compact sized matrix, say, $d = 16$, our method is much better than all other methods, in that the sum of the number of ciphertext-ciphertext multiplications and homomorphic automorphisms is at most 1/3 of the best existing algorithm, and the number of plaintext-ciphertext multiplications is at most 1/8 of the best existing algorithm.
2. For large sized matrix, as the size increases, the number of blocks of the matrix increases. If $d = 32$, we use $\ell = 4$ blocks; if $d = 64$, we use $\ell = 16$ blocks. Then the number of plaintext-ciphertext multiplications and homomorphic automorphisms increases by a factor of ℓ, and the number of ciphertext-ciphertext multiplications increases by a factor of $\ell^{\frac{1}{2}\log_2 7}$ when compared with the compact sized case.
3. The number of basic homomorphic operations in our algorithm is minimal, except for the case of $d = 64$ where the number of homomorphic automorphisms is bigger than that in [16]. This is because when $d \approx n^{\frac{1}{2}}$, our algorithm requires asymptotically $d^{2/3} \log d$ homomorphic automorphisms. In fact, we requires approximately $3d^{2/3} \log d$ homomorphic automorphisms,

while [16] requires only $2d + 4\sqrt{d} - 6$ homomorphic automorphisms. Therefore, for the specific value, $d = 64$, they require fewer homomorphic automorphisms than us.

4. Since we need the key-switching keys only to perform $\mathsf{HomTr}_{K/K_{12}}$, $\mathsf{HomTr}_{K/K_{13}}$ and $\mathsf{HomTr}_{K/K_{23}}$, for large sized matrix, the number of keys does not increase with the size of input matrix.
5. The work in [4,23,16] requires $m = m_1 m_2$, where m_1 and m_2 are coprime. Additionally, $n = \varphi(m_1) \cdot \varphi(m_2)$ must satisfy $d \mid \varphi(m_1)$ and $d \mid \varphi(m_2)$. In our new framework, there are no such restrictions.
6. The method in [25] only supports the special case such $d \leq \frac{1}{2}n^{\frac{1}{3}}$. If $d > \frac{1}{2}n^{\frac{1}{3}}$, for comparison purpose, we need to extend their method by homomorphic block matrix multiplication.

6 Implementation

Our implementation of the new framework and the algorithm therein is publicly available at

https://github.com/XiaopengZheng/HEMat.

It is based on the BGV scheme in HElib, a homomorphic encryption library. The experiment results are show in Table 3.

Table 3. Experiments on the new framework of homomophic matrix multiplication algorithm.

Size of matrix	Ring dimension n	Blocks	Plaintext modulus	Cipertext modulus	Runtime per	Security
$d \times d$	$r \times s \times t$	$L \times L$	p	$Q \cdot P$ (in bit)	HE-MatMult.	level
16×16	$16 \times 18 \times 20$	1×1	65537	151	0.158s	133.479
	$18 \times 18 \times 20$		$2^{31} - 1$	238	0.303s	87.4684
	$32 \times 18 \times 22$		$2^{31} - 1$	456	1.295s	87.9782
32×32	$16 \times 18 \times 20$	2×2	65537	151	0.231s	133.479
	$18 \times 18 \times 20$		$2^{31} - 1$	238	0.410s	87.4684
	$32 \times 18 \times 22$		$2^{31} - 1$	456	1.651s	87.9782
64×64	$16 \times 18 \times 20$	4×4	65537	151	0.691s	133.479
	$16 \times 18 \times 20$		$2^{31} - 1$	238	1.124s	87.4684
	$32 \times 18 \times 22$		$2^{31} - 1$	456	4.407s	87.9782

The following are some remarks on Table 3.

1. The level of security is $\lambda \geq 80$ bits. The security level is estimated by the program in HElib [13].
2. The ciphertext modulus contains the modulus used in key switching. As the ciphertext modulus increases, the security level decreases, so we need to increase the ring dimension to ensure sufficient security.

3. As the ciphertext modulus and the dimension of the ring increase, the time required for basic homomorphic operations also increases. Therefore, after selecting the ciphertext modulus based on the desired depth of the computation task, at the same time guaranteeing the desired security level, we choose a smaller value of n.
4. By Theorem 4.6, and also from the experimental result, the plaintext modulus p affects the growth of noise. In the experiments, a homomorphic matrix multiplication results in approximate 60 bits of loss in the ciphertext modulus for $p = 2^{31} - 1$, $n = 12672$, and $h = 2n/3$.
5. The number of threads used in our implementation is equal to the number of blocks.
6. The platform is a personal laptop with AMD Ryzen 7 6800H with Radeon Graphics of 64GB Memory running, and Ubuntu 22.04.2 LTS.

7 Conclusion and Further work

The paper presents a new framework for homomorphic matrix computations. It is shown to be more efficient compared with all existing methods, in that it requires fewer basic homomorphic operations and has lower communication cost.

In future studies, we will consider performing homomorphic multiplication of matrices with floating-point entries.

References

1. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014). https://doi.org/10.1145/2633600
2. Chen, H., Kim, M., Razenshteyn, I., Rotaru, D., Song, Y., Wagh, S.: Maliciously secure matrix multiplication with applications to private deep learning. In: Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III 26. pp. 31–59. Springer (2020). https://doi.org/10.1007/978-3-030-64840-4_2
3. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23. pp. 409–437. Springer (2017). https://doi.org/10.1007/978-3-319-70694-8_15
4. Cheon, J.H., Kim, A., Yhee, D.: Multi-dimensional packing for heaan for approximate matrix arithmetics. Cryptology ePrint Archive, Paper 2018/1245 (2018), https://eprint.iacr.org/2018/1245
5. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22. pp. 3–33. Springer (2016). https://doi.org/10.1007/978-3-662-53887-6_1

6. Costache, A., Smart, N.P.: Which ring based somewhat homomorphic encryption scheme is best? In: Topics in Cryptology-CT-RSA 2016: The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29-March 4, 2016, Proceedings. pp. 325–340. Springer (2016). https://doi.org/10.1007/978-3-319-29485-8_19
7. Ducas, L., Micciancio, D.: Fhew: bootstrapping homomorphic encryption in less than a second. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 617–640. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_24
8. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144 (2012), https://eprint.iacr.org/2012/144
9. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009). https://doi.org/10.1145/1536414.1536440
10. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. pp. 75–92. Springer (2013). https://doi.org/10.1007/978-3-642-40041-4_5
11. Halevi, S., Shoup, V.: Design and implementation of a homomorphic-encryption library. IBM Research (Manuscript) **6**(12-15), 8–36 (2013)
12. Halevi, S., Shoup, V.: Algorithms in helib. In: Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34. pp. 554–571. Springer (2014). https://doi.org/10.1007/978-3-662-44371-2_31
13. Halevi, S., Shoup, V.: Design and implementation of helib: a homomorphic encryption library. Cryptology ePrint Archive, Paper 2020/1481 (2020), https://eprint.iacr.org/2020/1481
14. Huang, H., Zong, H.: Secure matrix multiplication based on fully homomorphic encryption. The Journal of Supercomputing **79**(5), 5064–5085 (2023). https://doi.org/10.1007/s11227-022-04850-4
15. Huang, Z., Hong, C., Weng, C., Lu, W.j., Qu, H.: More efficient secure matrix multiplication for unbalanced recommender systems. IEEE Transactions on Dependable and Secure Computing **20**(01), 551–562 (2023). https://doi.org/10.1109/TDSC.2021.3139318
16. Jang, J., Lee, Y., Kim, A., Na, B., Yhee, D., Lee, B., Cheon, J.H., Yoon, S.: Privacy-preserving deep sequential model with matrix homomorphic encryption. In: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. pp. 377–391 (2022). https://doi.org/10.1145/3488932.3523253
17. Jiang, X., Kim, M., Lauter, K., Song, Y.: Secure outsourced matrix computation and application to neural networks. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. pp. 1209–1222 (2018). https://doi.org/https://doi.org/10.1145/3243734.3243837
18. Lee, Y., Micciancio, D., Kim, A., Choi, R., Deryabin, M., Eom, J., Yoo, D.: Efficient fhew bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 227–256. Springer (2023). https://doi.org/10.1007/978-3-031-30620-4_8

19. Liu, F.H., Wang, H.: Batch bootstrapping i: a new framework for simd bootstrapping in polynomial modulus. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 321–352. Springer (2023). https://doi.org/10.1007/978-3-031-30620-4_11
20. Lu, W., Kawasaki, S., Sakuma, J.: Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data. Cryptology ePrint Archive, Paper 2016/1163 (2016), https://eprint.iacr.org/2016/1163
21. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29. pp. 1–23. Springer (2010). https://doi.org/10.1007/978-3-642-13190-5_1
22. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-lwe cryptography. In: Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32. pp. 35–54. Springer (2013). https://doi.org/10.1007/978-3-642-38348-9_3
23. Rathee, D., Mishra, P.K., Yasuda, M.: Faster pca and linear regression through hypercubes in helib. In: Proceedings of the 2018 Workshop on Privacy in the Electronic Society. pp. 42–53 (2018). https://doi.org/10.1145/3267323.3268952
24. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) **56**(6), 1–40 (2009). https://doi.org/10.1145/1568318.1568324
25. Rizomiliotis, P., Triakosia, A.: On matrix multiplication with homomorphic encryption. In: Proceedings of the 2022 on Cloud Computing Security Workshop. pp. 53–61 (2022). https://doi.org/10.1145/3560810.3564267
26. Vinogradov, I.M.: Elements of number theory. Courier Corporation (2003)
27. Wang, S., Huang, H.: Secure outsourced computation of multiple matrix multiplication based on fully homomorphic encryption. KSII Transactions on Internet and Information Systems (TIIS) **13**(11), 5616–5630 (2019). https://doi.org/10.3837/tiis.2019.11.019