Designs, Codes and Cryptography (2025) 93:2671–2693
https://doi.org/10.1007/s10623-025-01614-y

# A new framework for fast homomorphic matrix multiplication

**Xiaopeng Zheng**[1] · **Hongbo Li**[2,3] · **Dingkang Wang**[2,3]

Received: 22 July 2024 / Revised: 7 February 2025 / Accepted: 3 March 2025 /
Published online: 15 March 2025


**Abstract**
Homomorphic encryption (HE) is one of the mainstream cryptographic tools used to enable secure outsourced computation. A typical task is secure matrix computation, which is a fundamental operation used in various outsourced computing applications such as statistical analysis and machine learning. In this paper, we present a new framework for secure multiplication of two matrices with size $r \times s$ and $s \times t$ respectively, which requires only $O(\log n)$ basic homomorphic operations if $rst \leq n$, where $n$ is dimension of the polynomial ring used in RLWE encryption. Our method was implemented in HElib using the BGV scheme. Experimental results show that the new framework has significant advantage in efficiency when $rst \leq n$. In this case, the new framework is 1.2 to 106.8 times faster than exiting algorithms in experiments.

**Keywords** Homomorphic encryption · Outsourced matrix multiplication · Tensor ring · Lattice basis and dual basis · Galois automorphism

**Mathematics Subject Classification** 11T71 · 94A60 · 68P25

## 1 Introduction

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

Communicated by C. Padro.

✉ Xiaopeng Zheng
xiaopengzheng@stu.edu.cn

Hongbo Li
hli@mmrc.iss.ac.cn

Dingkang Wang
dwang@mmrc.iss.ac.cn

[1] College of Mathematics and Computer Science, Shantou University, Shantou 515821, China

[2] State Key Laboratory of Mathematical Sciences, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China

[3] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

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) [13]. In 2014, Brakerski et al. [2] 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 [5, 6, 8, 12, 20, 21].

### 1.1 Secure non-square matrix multiplication based on HE

Secure matrix multiplication is a trending research topic due to its importance in secure data analysis and machine learning (e.g.,[18, 19, 26, 31]).. A naive approach to securely multiply two matrices of size $r \times s$ and $s \times t$ respectively is to use $rs+st$ distinct ciphertexts to represent these two matrices, which is very inefficient and requires a large amount of communication. Therefore, more efficient algorithms are proposed later on:

- [17] used a single ciphertext to represent a matrix, and introduced a new approach to homomorphically compute the multiplication of two matrices $A$ and $B$ with size $r \times s$ and $s \times s$ respectively, which requires $3r+5\sqrt{s}+\log(s/r)$ homomorphic automorphisms, $r$ ciphertext-ciphertext multiplications and $3s + 2r$ plaintext-ciphertext multiplications.
- [15] proposed an algorithm for secure matrix multiplication of matrices $A$ and $B$ with size $r \times s$ and $s \times t$. Compared to [17], this algorithm does not require the second matrix to be square. This new algorithm requires $s \log(\max\{s, t\}) + s$ homomorphic automorphisms, $s$ ciphertext-ciphertext multiplications and $s$ plaintext-ciphertext multiplications.
- Huang et al. [16] proposed a method of secure matrix multiplication by blockwise extracting the diagonals of one matrix and rotating the other matrix. For large size matrix, this method is sometimes better than other methods.
- [33] proposed another algorithm similar to that in [15]. Compared to [15], this algorithm requires less basic homomorphic operations including homomorphic automorphisms, ciphertext-ciphertext multiplications, and plaintext-ciphertext multiplications.
- Gao et al. [11] presents a novel fully homomorphic encryption scheme called GMS. For secure outsourced matrix multiplication matrices $A$ and $B$ with size $r \times s$ and $s \times t$, GMS only requires $O(\max\{m, n, l\})$ rotations and one homomorphic multiplication.
- Chen et al. [4] introduce a novel matrix encoding method, named bicyclic encoding, under which they propose two new algorithms BMM-I and BMM-II for encrypted matrix multiplication.

In Table 1, we summary the time complexity of the existing methods for secure multiplication for non-squaure matrices based on Homomorphic Encryption, which is counted by the number of basic homomorphic operations.

### 1.2 Our results

The current state-of-the-art HE schemes rely on the hardness of the Learning with Errors (LWE) problem or it ring variant (RLWE) [24, 27]. 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 and $p$ is prime. Let $n = \deg(\Phi_m(x))$. Generally, for security reasons, it is recommended that the value of $n$ should not be small. However, when $n$ significantly exceeds the size of matrices, the existing methods for secure matrix multiplication presented in [15, 16, 33] become inefficient. In this paper, we present a new framework for homomorphic matrix multiplication.

**Table 1** Number of basic homomorphic operations for matrix multiplication $A \times B$ with $A \in \mathbb{Z}_p^{r\times s}$ and $B \in \mathbb{Z}_p^{s\times t}$

| Method | Conditions | Cyclotomic Index $m$ | #CMult[a] | #Aut[b] | #Mult[c] | Depth[d] |
|---|---|---|---|---|---|---|
| [17] | $r \le s = t \le \left(\frac{n}{2}\right)^{\frac{1}{2}}$ | $2^l$ | $3s + 2r$ | $3r + 5\sqrt{s} + \log(s/r)$ | $r$ | 2 C + 1 M |
| [16] | $r \le s = t$ and $rs \le \left(\frac{n}{2}\right)^{\frac{1}{2}}$ | $2^l$ | $5s$ | $3s + \frac{6s}{\sqrt{r}} + \frac{s}{r}$ | $s$ | 2 C + 1 M |
| | $r = t \ge s$ and $rs \le \left(\frac{n}{2}\right)^{\frac{1}{2}}$ | | $5r$ | $3r + \frac{6r}{\sqrt{s}} + \frac{r}{s}$ | $r$ | 2 C + 1 M |
| [15][g] | $\max\{rs, st, rt\} \le n$ | $km_0m_1 + 1$ | $s$ | $s \log k_4 + s$ | $s$ | 1 C + 1 M |
| [33][g] | $\max\{rs, st, rt\} \le n$ | $km_0m_1 + 1$ | $2s$ | $4s + \log \frac{k_1^3}{k_2 k_3^2}$ | $k_3$ | 1 C + 1 M |
| [4][h] | $\max\{rs, st, rt\} \le \frac{n}{4}$ | $2^l$ | 0 | $2(s + \lceil \log \frac{r}{s} \rceil + \lceil \log \frac{t}{s} \rceil + 1)$ | $s$ | 1 M |
| | $r \cdot s \cdot t \le \frac{n}{2}$ | $2^l$ | 1 | $\lceil \log r \rceil + \lceil \log s \rceil + \lceil \log t \rceil$ | 1 | 1 C + 1 M |
| Ours[i] | $r \cdot s \cdot t \le n$ | $m_1m_2m_3$ | 0 | $\le 2\log(\varphi(m_2))$ | 1 | 1 M |

[a]CMult: plaintext-ciphertext multiplication;
[b]Aut: homomorphic automorphism;
[c]Mult: ciphertext-ciphertext multiplication;
[d]Depth: multiplication depth required for one matrix multiplicaton (since CMult and Mult introduce the noise significantly in HE scheme, we only count the depth of CMult and Mult similar to [15–17, 33]), where CMult is denoted by C, and Mult is denoted by M;
[e]$k_1 = \max\{r, s, t\}, k_2 = \text{median}\{r, s, t\}, k_3 = \min\{r, s, t\}; k_4 = \max\{s, t\}$;
[f]$n$: dimension of polynomial ring used in RLWE encryption
[g]$k, m_0, m_1$ are pairwise coprime and $r \le m_0, \max\{s, t\} \le m_1$
[h]$r, s, t$ are pairwise coprime
[i]$m_1, m_2, m_3$ are pairwise coprime and $r \le \varphi(m_1), s \le \varphi(m_2), t \le \varphi(m_3)$, where $\varphi$ is the Euler's totient function

It outperforms all other existing methods for secure multiplication of two matrices with size $r \times s$ and $s \times t$ respectively when $r \times s \times t \leq n$. Specifically, the main result is as follows:

**Theorem 1** *Suppose that $\mathcal{R}_p = \mathbb{Z}_p[x]/(\Phi_m(x))$ with $m = m_1 m_2 m_3$, where $m_1, m_2, m_3$ are pairwise coprime. Then the homomorphic matrix multiplication of matrices $A \in \mathbb{Z}_p^{r\times s}$ and $B \in \mathbb{Z}_p^{s\times t}$ requires only one ciphertext-ciphertext multiplication and at most $2\log(\varphi(m_2))$ homomorphic automorphisms, where $r \leq \varphi(m_1)$, $s \leq \varphi(m_2)$, $t \leq \varphi(m_3)$, and $\varphi$ is the Euler's totient function.*

Our implementation is publicly available at https://github.com/XiaopengZheng/HEMat_nonsquare

It is based on homomorphic encryption library–HElib. Experimental results show that the new framework has the best performance for secure matrix multiplication when $r \cdot s \cdot t \leq n$, for which case it is 4.0 to 106.8 times faster than existing algorithms (Table 2 in Sect. 5).

### 1.3 Our techniques

Let $\mathcal{R} = \mathbb{Z}[x]/(\Phi_m(x))$, where $\Phi_m(x)$ is the $m$-th cyclotomic polynomial with $\deg(\Phi_m(x)) = n$. We choose $m = m_1 m_2 m_3$, with factors $m_1$, $m_2$ 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 [21, 23], 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 each $\mathcal{R}_i^\vee$ is the dual of $\mathcal{R}_i$ for $i = 1, 2, 3$. Then $n = r \cdot s \cdot t$.

Suppose that $A = (a_{ij})$ and $B = (b_{ij})$ are two integer matrices modulus $p$ of size $r \times s$ and $s \times t$, respectively. We find that the matrix multiplication can be achieved by polynomial multiplication and trace operation. Specifically, let

$$\mathbf{m}_1 = \sum_{i=1}^{r}\sum_{j=1}^{s} a_{ij}\mathbf{u}_i\mathbf{v}_j \quad \text{and} \quad \mathbf{m}_2 = \sum_{i=1}^{s}\sum_{j=1}^{t} b_{ij}\mathbf{v}_i^\vee\mathbf{w}_j. \tag{1}$$

Then

$$\mathrm{Tr}(\mathbf{m}_1 \cdot \mathbf{m}_2) = \sum_{i=1}^{r}\sum_{j=1}^{t} c_{ij}\mathbf{u}_i\mathbf{w}_j, \tag{2}$$

with $(c_{ij}) = A \cdot B$ (see Theorem 2). This discovery allows us to implement matrix multiplication through polynomial operations. We summarize the new framework of homomorphic matrix multiplication in Fig. 1.

In [4, 11, 15–17, 33], the matrix multiplication result retains the same encoding format, enabling the algorithms to perform iterative matrix products. However, in our framework, the encoding format of the multiplication result changes, which is a limitation of our algorithm. Despite this, our algorithm is slightly better than the methods in [9, 22, 25], which are restricted to computing only a single multiplication between two matrices. In our algorithm, if the client wishes to perform repeated matrix multiplications, the matrices must be encoded in appropriate formats. Specifically, to multiply the matrices $A = (a_{ij}) \in \mathbb{Z}^{r\times s}$, $B = (b_{ij}) \in \mathbb{Z}^{s\times t}$, and $C = (c_{ij}) \in \mathbb{Z}^{t\times s}$, let

$$\mathbf{m}_A = \sum_{i=1}^{r}\sum_{j=1}^{s} a_{ij}\mathbf{u}_i\mathbf{v}_j, \quad \mathbf{m}_B = \sum_{i=1}^{s}\sum_{j=1}^{t} b_{ij}\mathbf{v}_i^\vee\mathbf{w}_j, \text{ and } \quad \mathbf{m}_C = \sum_{i=1}^{t}\sum_{j=1}^{s} c_{ij}\mathbf{w}_i^\vee\mathbf{v}_j.$$

**Table 2** Comparison with Previous Work

| Size of matrices | Method | $m$ | $n$ | $Q^{a}$ | Time (s) | Our advantage[b] |
|---|---|---|---|---|---|---|
| $A_{5\times100} \times B_{100\times20}$ | [15] | $101 \cdot 125$ | 10000 | 252 | 25.4197 | 106.8× |
| | [33] | $101 \cdot 125$ | 10000 | 252 | 7.0491 | 29.6× |
| | [4]-I | 32768 | 16384 | 165 | 6.7653 | 28.4× |
| | [4]-II | 65536 | 32768 | 176 | 0.5341 | 2.2× |
| | Ours | $7 \cdot 101 \cdot 23$ | $6 \cdot 100 \cdot 22$ | 144 | **0.2381** | 1× |
| $A_{5\times20} \times B_{20\times100}$ | [15] | $101 \cdot 125$ | 10000 | 252 | 5.1911 | 27.4× |
| | [33] | $101 \cdot 125$ | 10000 | 252 | 1.4919 | 7.9× |
| | [4]-I | 32768 | 16384 | 165 | 1.4860 | 7.9× |
| | [4]-II | 65536 | 32768 | 176 | 0.5332 | 2.8× |
| | Ours | $7 \cdot 23 \cdot 101$ | $6 \cdot 22 \cdot 100$ | 144 | **0.1892** | 1× |
| $A_{100\times5} \times B_{5\times20}$ | [15] | $101 \cdot 125$ | 10000 | 252 | 1.1554 | 6.8× |
| | [33] | $101 \cdot 125$ | 10000 | 252 | 1.0109 | 5.8× |
| | [4]-I | 32768 | 16384 | 165 | 0.5763 | 3.3× |
| | [4]-II | 65536 | 32768 | 176 | 0.5390 | 3.0× |
| | Ours | $101 \cdot 7 \cdot 23$ | $100 \cdot 6 \cdot 22$ | 144 | **0.1769** | 1× |
| $A_{50\times4} \times B_{4\times50}$ | [16] | 16384 | 8192 | 246 | 3.7524 | 26.5× |
| | [15] | $101 \cdot 125$ | 10000 | 252 | 1.2434 | 11.7× |
| | [33] | $101 \cdot 125$ | 10000 | 252 | 1.0208 | 9.6× |
| | [4]-I | 32768 | 16384 | 165 | 0.5080 | 4.8× |
| | [4]-II | 65536 | 32768 | 176 | 0.5008 | 4.7× |
| | Ours | $53 \cdot 5 \cdot 59$ | $52 \cdot 4 \cdot 58$ | 144 | **0.1065** | 1× |
| $A_{4\times50} \times B_{50\times50}$ | [16] | 16384 | 8192 | 246 | 3.7596 | 16.9× |
| | [15] | $101 \cdot 125$ | 10000 | 252 | 11.0501 | 41.8× |
| | [33] | $101 \cdot 125$ | 10000 | 252 | 2.9371 | 11.1× |
| | [17] | 32768 | 16384 | 246 | 0.9021 | 4.0× |
| | [4]-I | 32768 | 16384 | 165 | 3.4560 | 15.6× |
| | [4]-II | 65536 | 32768 | 176 | 0.4952 | 2.2× |
| | Ours | $5 \cdot 53 \cdot 59$ | $4 \cdot 52 \cdot 58$ | 144 | **0.2219** | 1× |
| $A_{2\times50} \times B_{50\times50}$ | [16] | 16384 | 8192 | 246 | 1.9912 | 15.8× |
| | [15] | $101 \cdot 125$ | 10000 | 252 | 10.9887 | 87.1× |
| | [33] | $101 \cdot 125$ | 10000 | 252 | 2.7526 | 21.8× |
| | [17] | 16384 | 8192 | 246 | 0.8168 | 6.5× |
| | [4]-I | 32768 | 16384 | 165 | 3.4299 | 27.2× |
| | [4]-II | 32768 | 16384 | 176 | 0.2203 | 1.7× |
| | Ours | $53 \cdot 3 \cdot 59$ | $52 \cdot 2 \cdot 58$ | 144 | **0.1261** | 1× |
| $A_{1\times100} \times B_{100\times100}$ | [16] | 16384 | 8192 | 246 | 2.1305 | 5.2× |
| | [15] | $101 \cdot 125$ | 10000 | 252 | 26.0663 | 64.2× |
| | [33] | $101 \cdot 125$ | 10000 | 252 | 5.3727 | 13.4× |
| | [17] | $4 \cdot 128 \cdot 128$ | 32768 | 244 | 5.6998 | 14.0× |
| | [4]-I | 131072 | 65536 | 165 | 36.5530 | 90.0× |

**Table 2** continued

| Size of matrices | Method | $m$ | $n$ | $Q^{a}$ | Time (s) | Our advantage[b] |
|---|---|---|---|---|---|---|
| | [4]-II | 65536 | 32768 | 176 | 0.4996 | 1.2× |
| | Ours | $2 \cdot 101 \cdot 103$ | $1 \cdot 100 \cdot 102$ | 144 | **0.4063** | 1× |

[a] $Q$: the number of bits of the cyphertext modulus
[b] ratio of time cost by the list method over our method
[c] The modulus of plaintext is about 32 bits

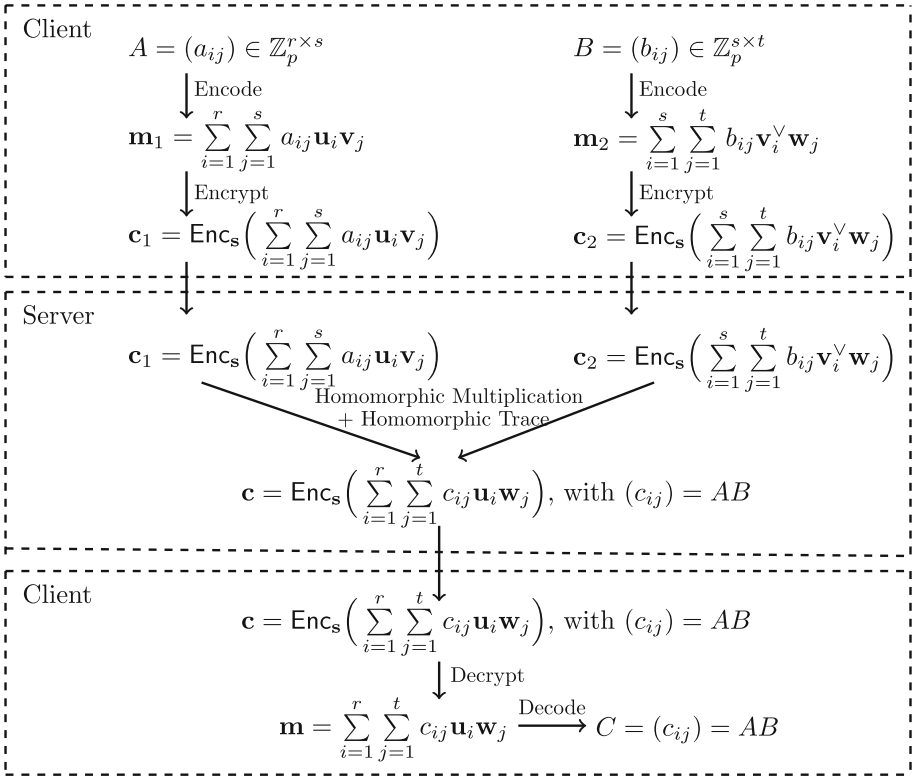


**Fig. 1** Framework for Secure Matrix Multiplication of matrices $A \in \mathbb{Z}_p^{r\times s}$ and $B \in \mathbb{Z}_p^{s\times t}$

We then have

$$\mathrm{Tr}_{K/K_{12}}(\mathrm{Tr}_{K/K_{13}}(\mathbf{m}_A \cdot \mathbf{m}_B) \cdot \mathbf{m}_C) = \sum_{i=1}^{r}\sum_{i=1}^{s} d_{ij}\mathbf{u}_i\mathbf{v}_j,$$

where $(d_{ij}) = A \cdot B \cdot C$ ($K$, $K_{12}$ and $K_{13}$ will be defined in Sect. 3.1).

### 1.4 Organization

The rest of the paper is organized as follows. In Sect. 2, we provide the necessary background on algebraic number theory and BGV homomorphic encryption scheme. In Sect. 3, we present a new framework for plaintext matrix multiplication via tensor ring. In Sect. 4, we extend the

framework in Sect. 3 to the homomorphic case. Section 5 provides more detailed comparisons with existing methods under specific parameters. Section 6 provides the conclusion.

## 2 Preliminaries

### 2.1 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 $\{0, 1, ..., k-1\}$.

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 font, 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 $Q$.

### 2.2 Algebraic number theory background

We present some necessary background of algebraic number theory. Further details can be found in reference [21, 23].

#### 2.2.1 Algebraic number field

An Algebraic number field is an algebraic field extension of $\mathbb{Q}$ expressed as $K = \mathbb{Q}(\alpha)$ by adjoining $\alpha$ to $\mathbb{Q}$, where $\alpha$ is a root of an irreducible polynomial $f(x)$ in $\mathbb{Z}[x]$. Let $\xi_m$ 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 $m$.

#### 2.2.2 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 $K$.

#### 2.2.3 Trace

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

$$\mathrm{Tr}_{K/k}(\mathbf{a}) = \sum_{\sigma \in \mathrm{Gal}(K/k)} \sigma(\mathbf{a}). \tag{3}$$

### 2.2.4 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:

$$\mathcal{R}^\vee = \{\mathbf{a} \in K : \mathrm{Tr}(x) \in \mathbb{Z} \text{ for all } x \in a\mathcal{R}\}. \tag{4}$$

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

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

Further details on the dual basis can be found in [23].

***Example 1*** Let $K = \mathbb{Q}[x]/(x^8 + 1)$ and $\mathcal{R} = \mathbb{Z}[x]/(x^8 + 1)$. Then $\mathbf{B} = \{1, x, x^2, x^3, x^4, x^5, x^6, x^7\}$ is a $\mathbb{Z}$-basis for $\mathcal{R}$. And

$$\mathbf{B}^\vee = \left(\frac{1}{8}, -\frac{x^7}{8}, -\frac{x^6}{8}, -\frac{x^5}{8}, -\frac{x^4}{8}, -\frac{x^3}{8}, -\frac{x^2}{8}, -\frac{x}{8}\right).$$

is a dual basis of $\mathbf{B}$.

### 2.2.5 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 fields:

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

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 indeterminate $x_i$ and modulus polynomial $\Phi_{m_i}(x_i)$ 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 $\mathcal{R}^\vee$.

## 2.3 Homomorphic encryption

### 2.3.1 Ring learning with errors

Let $\chi$ be a distribution over $\mathcal{R}_Q$ and $\sigma > 0$ be a real. Let $n = \deg(\Phi_m(x))$. The ring learning with errors (RLWE) assumption with respect to the parameters $(n, Q, \chi, \sigma)$ is the following:

for a fixed $\mathbf{s}_0 \leftarrow \chi$, given polynomially many samples of either $(\mathbf{a}, \mathbf{b})$ or $(\mathbf{a}, \mathbf{a}\mathbf{s}_0 + \mathbf{e})$, where $\mathbf{a}, \mathbf{b} \leftarrow \mathcal{R}_Q, \mathbf{e} \leftarrow \mathcal{DG}(\sigma^2)$, it is computationally hard to distinguish between the two groups of samples, where $\mathcal{DG}(\sigma^2)$ the discrete Guassian distribution of variance $\sigma^2$. The most popular HE schemes such as FV [10], BGV [2] and CKKS [5] rely on the security provided by the RLWE assumption. In practical, $\chi$ is chosen to be $\mathcal{HWT}(h)$ with some positive integer $h$, where $\mathcal{HWT}(h)$ is the set of signed binary vectors in $\{0, \pm 1\}^n$ whose Hamming weight is exactly $h$.

### 2.3.2 BGV scheme [2]

The plaintext space and ciphertext space are $\mathcal{R}_p$ and $\{\mathcal{R}_{Q_i} : i = 1, \ldots, \ell\}$ respectively.

- $\mathsf{BGV.Setup}(1^\lambda, p, m, S)$: For given security parameter $\lambda$, 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 decreasing ciphertext moduli $\{Q_i, i = 1, \ldots, \ell\}$, a modulus $P$, a key distribution $\mathcal{HWT}(h)$, and an error parameter $\sigma$. The public parameters are $\mathsf{pp} = (m, p, Q_1, \ldots, Q_\ell, P, \chi, \sigma, S)$, $i = 1, \ldots, \ell$.
- $\mathsf{BGV.KeyGen}(\mathsf{pp})$: Given public parameters $\mathsf{pp}$, generate a secret key $\mathbf{s} = (-\mathbf{s_0}, 1)$ with $\mathbf{s_0} \leftarrow \mathcal{HWT}(h)$, a public key $\mathsf{pk}$, a relinearization key $\mathsf{rlk}$, and automorphism (key-switching) keys $\{\mathsf{atk}_i : i \in S\}$ of the automorphisms $\mathbf{m}(x) \mapsto \mathbf{m}(x^i)$ for all $i \in S$.
- $\mathsf{BGV.Enc}(\mathsf{pk}, \mathbf{m})$: Given a public key $\mathsf{pk}$ and a plaintext $\mathbf{m} \in \mathcal{R}_p$, output a ciphertext $\mathbf{c}$ encrypting plaintext $\mathbf{m}$.
- $\mathsf{BGV.Dec}(\mathbf{s}, \mathbf{c})$: Given a ciphertext $\mathbf{c}$ with secret key $\mathbf{s}$, output the encrypted plaintext $\mathbf{m} \in \mathcal{R}_p$.
- $\mathsf{BGV.Add}(\mathbf{c}, \mathbf{c}')$: Given two ciphertexts $\mathbf{c}$ and $\mathbf{c}'$ encrypting plaintexts $\mathbf{m}_1$ and $\mathbf{m}_2$ respectively, output a ciphertext $\mathbf{c}_{\text{add}}$ encrypting plaintext $\mathbf{m}_1 + \mathbf{m}_2$.
- $\mathsf{BGV.Mult}(\mathsf{rlk}, \mathbf{c}, \mathbf{c}')$: Given two ciphertexts $\mathbf{c}$ and $\mathbf{c}'$ encrypting plaintexts $\mathbf{m}_1$ and $\mathbf{m}_2$ respectively and a relinearization key $\mathsf{rlk}$, output a ciphertext $\mathbf{c}_{\text{mul}}$ encrypting plaintext $\mathbf{m}_1\mathbf{m}_2$.
- $\mathsf{BGV.CMult}(\mathbf{c}, \mathbf{f})$: Given a ciphertexts $\mathbf{c}$ encrypting plaintext $\mathbf{m}$, and given a plaintext $\mathbf{f} \in \mathcal{R}_p$, output a ciphertext $\mathbf{c}_{\text{cmul}}$ encrypting plaintext $\mathbf{f} \cdot \mathbf{m}$.
- $\mathsf{BGV.Auto}(\mathsf{atk}_i, \mathbf{c})$: Given a ciphertext $\mathbf{c}$ encrypting plaintext $\mathbf{m}(x)$ and an automorphism key $\mathsf{atk}_i$ for some $i \in S$, output a ciphertext $\mathbf{c}_{\text{aut}}$ encrypting plaintext $\mathbf{m}(x^i)$ with secret key $\mathbf{s}$.
- $\mathsf{BGV.ModulusSwitch}(\mathbf{c}, Q_i, Q_j)$: Given a ciphertext $\mathbf{c} \in \mathcal{R}_{Q_i}^2$ encrypting plaintext $\mathbf{m}$ 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 same plaintext $\mathbf{m}$.

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

### 2.3.3 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 $\xi^j$, where $j \in \mathbb{Z}_m^*$, since $\xi^j$ 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:

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

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

### 2.3.4 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 $\|\mathbf{e}\|_{\text{can}} \le E$. We will recall the estimate of noise in the BGV scheme [7, 14, 28].

**Lemma 1** *(Error estimation of* BGV.ModulusSwitch*) 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* 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*

$$B_{\text{scale}} = p\left(\sqrt{3n} + 8\sqrt{nh/3}\right) \tag{5}$$

*is the bound of rounding error.*

**Lemma 2** *(Error estimation of* BGV.Auto*) 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* $\mathbb{Z}_m^*$ *and let the gadget vector* $\vec{g}$ *used in key-switching be* $(1, 2, \ldots, 2^{\lfloor \log Q\rfloor})$*. Then* BGV.Auto$(\text{atk}_i, \mathbf{c})$ *outputs a ciphertext* $\mathbf{c}'$ *encrypting* $\mathbf{m}(x^i) \in \mathcal{R}_p$ *with secret key* $\mathbf{s}$*, and the noise of* $\mathbf{c}'$ *is bounded by* $E + B_{\text{ks}}$*, where*

$$B_{\text{ks}} = \frac{16}{\sqrt{3}} pn\sigma \lceil \log Q\rceil \tag{6}$$

*is the bound of key-switching error.*

**Lemma 3** *(Error estimation of* BGV.CMult*) 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* 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 4** *(Error estimation of* BGV.Mult*) For* $l = 1, 2$*, 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*

$$q \approx \min\left\{(B_{\text{scale}}\cdot Q_1)/E_1, (B_{\text{scale}}\cdot Q_2)/E_2\right\}, \tag{7}$$

*where* $B_{\text{scale}}$ *in given in* (5)*. Then* BGV.Mult$(\text{rlk}, \mathbf{c}_1, \mathbf{c}_2)$ *outputs a ciphertext in* $\mathcal{R}_q^2$ *encrypting* $\mathbf{m}_1\cdot\mathbf{m}_2$ *with noise bounded by* $\left((q/Q_1)E_1 + B_{\text{scale}}\right)\left((q/Q_2)E_2 + B_{\text{scale}}\right) + B_{\text{ks}}$*, where* $B_{\text{ks}}$ *is given in* (6)*.*

We emphasize that to choose a proper common ciphertext modulus $q$, in HElib [28], 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 the new framework, we choose $m = m_1m_2m_3$ with $m_1$, $m_2$ and $m_3$ pairwise coprime. In Sect. 2.2, we have 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 $\mathcal{R} \cong \mathcal{R}_1 \otimes \mathcal{R}_2 \otimes \mathcal{R}_3$. We introduce the following notations similar to those in [21]:

- $K_{12}$, $K_{13}$ 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.
- $\{\mathbf{u}_1, \ldots, \mathbf{u}_r\}$, $\{\mathbf{v}_1, \ldots, \mathbf{v}_s\}$ and $\{\mathbf{w}_1, \ldots, \mathbf{w}_t\}$ denote the $\mathbb{Z}$-bases of $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$, respectively.
- $\{\mathbf{u}_1^\vee, \ldots, \mathbf{u}_r^\vee\}$, $\{\mathbf{v}_1^\vee, \ldots, \mathbf{v}_s^\vee\}$ and $\{\mathbf{w}_1^\vee, \ldots, \mathbf{w}_t^\vee\}$ 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 below will be used in this paper:

- $n = \varphi(m_1) \cdot \varphi(m_2) \cdot \varphi(m_3)$ is the dimension of ring $\mathcal{R}$.
- For $A = (a_{ij}) \in \mathbb{Z}^{r\times s}$, $B = (b_{ij}) \in \mathbb{Z}^{s\times t}$, $C = (c_{ij}) \in \mathbb{Z}^{r\times t}$ with $r \le \varphi(m_1)$, $s \le \varphi(m_2)$ and $t \le \varphi(m_3)$, denote

$$\mathbf{m}_A^{(\mathbf{uv})} = \sum_{i=1}^{r}\sum_{j=1}^{s} a_{ij}\mathbf{u}_i\mathbf{v}_j, \quad \mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})} = \sum_{i=1}^{s}\sum_{j=1}^{t} b_{ij}\mathbf{v}_i^\vee\mathbf{w}_j, \quad \mathbf{m}_C^{(\mathbf{uw})} = \sum_{i=1}^{r}\sum_{j=1}^{t} c_{ij}\mathbf{u}_i\mathbf{w}_j.$$

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

### 3.2 Matrix multiplication via polynomial operations

Suppose that $r \le \varphi(m_1)$, $s \le \varphi(m_2)$ and $t \le \varphi(m_3)$. For input matrices $A \in \mathbb{Z}^{r\times s}$ and $B \in \mathbb{Z}^{s\times t}$, to compute $AB$, we encode them respectively as:

$$\begin{aligned} A = (a_{ij}) \in \mathbb{Z}^{r\times s} &\xrightarrow{\text{Encode}} \quad \mathbf{m}_A^{(\mathbf{uv})} = \sum_{i=1}^{r}\sum_{j=1}^{s} a_{ij}\mathbf{u}_i\mathbf{v}_j, \\ B = (b_{ij}) \in \mathbb{Z}^{s\times t} &\xrightarrow{\text{Encode}} \quad \mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})} = \sum_{k=1}^{s}\sum_{l=1}^{t} b_{kl}\mathbf{v}_k^\vee\mathbf{w}_l. \end{aligned} \tag{8}$$

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 2** *Let $A = (a_{ij}) \in \mathbb{Z}^{r\times s}$ and $B = (b_{ij}) \in \mathbb{Z}^{s\times t}$ be two matrices. Then*

$$\mathrm{Tr}_{K/K_{13}}(\mathbf{m}_A^{(\mathbf{uv})} \cdot \mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})}) = \sum_{i=1}^{r}\sum_{l=1}^{t} c_{il}\mathbf{u}_i\mathbf{w}_l, \tag{9}$$

*with $(c_{ij})_{1\le i,j\le d} = A \cdot B$.*

***Proof*** For all $i, j = 1, \ldots, d$, since $\mathbf{u}_i$ and $\mathbf{w}_j$ both belong to $K_{13}$, $\sigma(\mathbf{u}_i) = \mathbf{u}_i$ and $\sigma(\mathbf{w}_j) = \mathbf{w}_j$ for any $\sigma \in \mathrm{Gal}(K/K_{13})$. So

$$\begin{aligned} \mathrm{Tr}_{K/K_{13}}\left(\mathbf{m}_A^{(\mathbf{uv})} \cdot \mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})}\right) &= \mathrm{Tr}_{K/K_{13}}\left(\sum_{i,j,k,l} a_{ij}b_{kl}\mathbf{u}_i\mathbf{v}_j\mathbf{v}_k^\vee\mathbf{w}_l\right) \\ &= \sum_{i,j,k,l} a_{ij}b_{kl}\mathbf{u}_i\left(\sum_{\sigma\in\mathrm{Gal}(K/K_{13})}\sigma(\mathbf{v}_j\mathbf{v}_k^\vee)\right)\mathbf{w}_l. \end{aligned} \tag{10}$$

Notice that $\mathrm{Gal}(K/K_{13})$ is isomorphic to $\mathrm{Gal}(K_2/\mathbb{Q})$ by the restriction map $\sigma \mapsto \sigma|_{K_2}$ for all $\sigma \in \mathrm{Gal}(K/K_{13})$. Since $\mathbf{v}_j\mathbf{v}_k^\vee \in K_2$, we have

$$\begin{aligned}\sum_{\sigma\in\mathrm{Gal}(K/K_{13})} \sigma(\mathbf{v}_j\mathbf{v}_k^\vee) &= \sum_{\sigma\in\mathrm{Gal}(K_2/\mathbb{Q})} \sigma(\mathbf{v}_j\mathbf{v}_k^\vee) \\ &= \mathrm{Tr}_{K_2/\mathbb{Q}}(\mathbf{v}_j\mathbf{v}_k^\vee) = \begin{cases}1, & j=k,\\ 0, & j\neq k.\end{cases}\end{aligned} \tag{11}$$

Substituting it into (10), we get

$$\mathrm{Tr}_{K/K_{13}}\left(\mathbf{m}_A^{(\mathbf{uv})}\cdot\mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})}\right) = \sum_{i=1}^{r}\sum_{l=1}^{t}\left(\sum_{j=1}^{s} a_{ij}b_{jl}\right)\mathbf{u}_i\mathbf{w}_l = \sum_{i=1}^{r}\sum_{l=1}^{t} c_{il}\mathbf{u}_i\mathbf{w}_l,$$

where $(c_{il}) = A\cdot B$. □

***Example 2*** Let $m_1 = 5$, $m_2 = 4$ and $m_3 = 3$. Then $\Phi_5(x) = x^4+x^3+x^2+x+1$, $\Phi_4(y) = y^2+1$, $\Phi_3(z) = z^2+z+1$, and

$$r = \varphi(m_1) = 4,\ s = \varphi(m_2) = 2 \text{ and } t = \varphi(m_3) = 2.$$

Let $K_1 = \mathbb{Q}[x]/(\Phi_5(x))$, $K_2 = \mathbb{Q}[y]/(\Phi_4(y))$, and $K_3 = \mathbb{Q}[z]/(\Phi_3(z))$. Let $K = \mathbb{Q}[x]/(\Phi_{60}(x))$, then

$$K \cong \mathbb{Q}[x,y,z]/(\Phi_5(x),\Phi_4(y),\Phi_3(z)).$$

All computations in this example are conducted in $K$, i.e., mod $(\Phi_5(x),\Phi_4(y),\Phi_3(z))$. Let $\mathcal{R}_1 = \mathbb{Z}[x]/(\Phi_5(x))$, $\mathcal{R}_2 = \mathbb{Z}[y]/(\Phi_4(y))$ and $\mathcal{R}_3 = \mathbb{Z}[z]/(\Phi_3(z))$. Compute the $\mathbb{Z}$-bases of $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, and the dual basis in $\mathcal{R}_2^\vee$:

$$\begin{aligned}\{\mathbf{u}_1,\mathbf{u}_2,\mathbf{u}_3,\mathbf{u}_4\} &= \{1,x,x^2,x^3\}, \qquad \{\mathbf{v}_1,\mathbf{v}_2\} = \{1,y\},\\ \{\mathbf{v}_1^\vee,\mathbf{v}_2^\vee\} &= \left\{\frac{1}{2},-\frac{y}{2}\right\}, \qquad \{\mathbf{w}_1,\mathbf{w}_2\} = \{1,z\}.\end{aligned}$$

Suppose that $A = \begin{pmatrix}1&2\\3&4\\5&6\\7&8\end{pmatrix}$ and $B = \begin{pmatrix}5&6\\7&8\end{pmatrix}$. Then we encode $A$ as

$$\begin{aligned}\mathbf{m}_A^{(\mathbf{uv})} &= \mathbf{u}_1\mathbf{v}_1 + 2\mathbf{u}_1\mathbf{v}_2 + 3\mathbf{u}_2\mathbf{v}_1 + 4\mathbf{u}_2\mathbf{v}_2 + 5\mathbf{u}_3\mathbf{v}_1 + 6\mathbf{u}_3\mathbf{v}_2\\ &\quad + 7\mathbf{u}_4\mathbf{v}_1 + 8\mathbf{u}_4\mathbf{v}_2\\ &= 1 + 2y + 3x + 4xy + 5x^2 + 6x^2y + 7x^3 + 8x^3y.\end{aligned}$$

Similarly, we encode $B$ as

$$\begin{aligned}\mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})} &= 5\mathbf{v}_1^\vee\mathbf{w}_1 + 6\mathbf{v}_1^\vee\mathbf{w}_2 + 7\mathbf{v}_2^\vee\mathbf{w}_1 + 8\mathbf{v}_2^\vee\mathbf{w}_2\\ &= \frac{5}{2} + \frac{6}{2}z - \frac{7}{2}y - \frac{8}{2}yz.\end{aligned}$$

Let $\mathbf{m}(x,y,z)$ be the notation for $\mathbf{m}_A^{(\mathbf{uv})}\mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})}$. The trace of $\mathbf{m}$ is computed as follows:

$$\mathrm{Tr}_{K/K_{12}}(\mathbf{m}) = \sum_{\sigma\in\mathrm{Gal}(K/K_{12})}\sigma(\mathbf{m})$$

$$\begin{aligned} &= \mathbf{m}(x, y, z) + \mathbf{m}(x, y^2, z) \\ &= 19 + 22z + 43x + 50xz + 67x^2 + 78x^2z + 91x^3 + 106x^3z, \end{aligned}$$

which corresponds to the matrix

$$C = \begin{pmatrix} 19 & 22 \\ 43 & 50 \\ 67 & 78 \\ 91 & 106 \end{pmatrix} = A \cdot B.$$

### 3.3 Fast trace computation

Let $K/F$ be a Galois extension with $r = |\mathrm{Gal}(K/F)|$. The direct computation of the trace, by definition, requires $r-1$ automorphisms. It is well-known that the Galois group of cyclotomic extension $\mathbb{Q}(\xi_m)$ over $\mathbb{Q}$ is isomorphic to the multiplicative group $\mathbb{Z}_m^*$. This group is cyclic if and only if $m$ is 1, 2, 4, $p^k$, or $2p^k$, where $p$ is an odd prime and $k > 0$ [29]. We first present an algorithm for computing the trace function for cyclic Galois groups, which requires only $O(\log r)$ automorphisms.

**Proposition 1** *Let $K/F$ be a Galois extension, where $\mathrm{Gal}(K/F)$ is cyclic. Let $\sigma$ be a generator of $\mathrm{Gal}(K/F)$. Then for any $\mathbf{m} \in K$ and any $1 \le \ell \le r = |\mathrm{Gal}(K/F)|$, $\sum_{i=0}^{\ell-1} \sigma^i(\mathbf{m})$ 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 $\ell \le k-1$. For $\ell = k$, if $\ell$ is even, then

$$\sum_{i=0}^{\ell-1} \sigma^i(\mathbf{m}) = \sum_{i=0}^{\ell/2-1} \sigma^i(\mathbf{m}) + \sigma^{\ell/2}\left(\sum_{i=0}^{\ell/2-1} \sigma^i(\mathbf{m})\right). \tag{12}$$

According to the induction 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 (12), $\sum_{i=0}^{\ell-1} \sigma^i(\mathbf{m})$ 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 $\ell$ is odd, then

$$\sum_{i=0}^{\ell-1} \sigma^i(\mathbf{m}) = \sum_{i=0}^{\frac{\ell-1}{2}-1} \sigma^i(\mathbf{m}) + \sigma^{\frac{\ell-1}{2}}\left(\sum_{i=0}^{\frac{\ell-1}{2}-1} \sigma^i(\mathbf{m})\right) + \sigma^{\ell-1}(\mathbf{m}). \tag{13}$$

According to the induction 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 (13), $\sum_{i=0}^{\ell-1} \sigma^i(\mathbf{m})$ 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 1, we get the following Algorithm 1 for trace computation.

***Remark 1*** If $m = 2^l$ with $l \ge 3$, then $r = \varphi(m) = 2^{l-1}$, and $\mathrm{Gal}(\mathbb{Q}(\xi_m)/\mathbb{Q}) \simeq \langle\sigma\rangle \times \langle\tau\rangle$, where $\sigma^{r/2} = 1$ and $\tau^2 = 1$. We can first invoke Algorithm 1 to compute $\mathbf{m} := \sum_{i=0}^{r/2-1} \sigma^i(\mathbf{m})$, and then compute $\mathbf{m} := \mathbf{m} + \tau(\mathbf{m})$ to obtain the trace of $\mathbf{m}$. The number of homomorphic automorphisms and additions is both $\log r$.

**Algorithm 1** $\mathsf{CircTr}(\mathbf{m}, \sigma, r)$

**Input:** A polynomial $\mathbf{m} \in \mathcal{R}$, the automophism $\sigma$, and an integer $r$.
**Output:** A polynomial $\sum_{i=0}^{r-1} \sigma^i(\mathbf{m})$.
1: **if** $r = 1$ **then**
2: **return** $\mathbf{m}$;
3: **end if**
4: **if** $r$ is even **then**
5: $\mathbf{m}_{\frac{r}{2}} = \mathsf{CircTr}(\mathbf{m}, \sigma, \frac{r}{2})$;
6: **return** $\mathbf{m}_{\frac{r}{2}} + \sigma^{\frac{r}{2}}(\mathbf{m}_{\frac{r}{2}})$;
7: **else**
8: $\mathbf{m}_{\frac{r-1}{2}} = \mathsf{CircTr}(\mathbf{m}, \sigma, \frac{r-1}{2})$;
9: **return** $\mathbf{m}_{\frac{r-1}{2}} + \sigma^{\frac{r-1}{2}}(\mathbf{m}_{\frac{r-1}{2}}) + \sigma^{r-1}(\mathbf{m})$;
10: **end if**

Proposition 1 proposes an algorithm for computing trace evaluations when the Galois group is cyclic, which was first introduced in [32]. Another case arises when the extension $\mathbb{Q}(\xi_m)/\mathbb{Q}$ exhibits a tower structure, which is widely used in [1, 3, 21]. Recently, Xia et al. [30] combined these two methods, enabling the generalization of the result in Proposition 1 to an arbitrary cyclotomic field extension.

**Proposition 2** [30] *Let $K = \mathbb{Q}(\xi_m)$, then $\mathrm{Tr}_{K/\mathbb{Q}}$ can be calculated using at most $2\log(\varphi(m))$ additions and $2\log(\varphi(m))$ automorphisms.*

The main idea of Proposition 2 is to construct a tower of fields, and apply Proposition 1 recursively. Suppose that $m = 2^{s_0} p_1^{s_1} \cdots p_l^{s_l}$, where $p_1, \ldots, p_l$ are odd prime numbers and pairwise coprime. Consider a tower of fields

$$\mathbb{Q} \subset \mathbb{Q}(\xi_{2^{s_0}}) \subset \mathbb{Q}(\xi_{2^{s_0}}, \xi_{p_1^{s_1}}) \subset \cdots \subset \mathbb{Q}(\xi_{2^{s_0}}, \xi_{p_1^{s_1}}, \cdots, \xi_{p_l^{s_l}}) = \mathbb{Q}(\xi_m).$$

Let $F_i = \mathbb{Q}(\xi_{2^{s_0}}, \xi_{p_1^{s_1}}, \cdots, \xi_{p_i^{s_i}})$, for $i = 0, 1, \ldots, l$. The trace can be decomposed as

$$\mathrm{Tr}_{K/\mathbb{Q}} = \mathrm{Tr}_{F_0/\mathbb{Q}} \circ \mathrm{Tr}_{F_1/F_0} \circ \cdots \circ \mathrm{Tr}_{F_l/F_{l-1}}.$$

Since $\mathrm{Gal}(F_i/F_{i-1}) \simeq \mathrm{Gal}(\mathbb{Q}(\xi_{p_i^{s_i}})/\mathbb{Q})$ is cyclic, by Proposition 1, the computation of each trace $\mathrm{Tr}_{F_i/F_{i-1}}$ requires at most $2\log(\varphi(p_i^{s_i}))$ automorphisms and $2\log(\varphi(p_i^{s_i}))$ additions. If $s_0 \leq 2$, then $\mathrm{Gal}(F_0/\mathbb{Q})$ is cyclic as well. Otherwise, by Remark 1, the computation of $\mathrm{Tr}_{F_0/\mathbb{Q}}$ requires $\log(\varphi(2^{s_0}))$ automorphisms and $\log(\varphi(2^{s_0}))$ additions. Therefore, the trace $\mathrm{Tr}_{K/\mathbb{Q}}$ can be calculated using at most

$$\log(\varphi(2^{s_0})) + 2\log(\varphi(p_1^{s_1})) + \cdots + 2\log(\varphi(p_l^{s_l})) \leq 2\log(\varphi(m))$$

automorphisms and $2\log(\varphi(m))$ additions.

Based on the explanation above, we can obtain an algorithm to compute the trace operator for an arbitrary cyclotomic field extension, which is presented in Algorithm 2.

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, then by Proposition 2 guarantees that $\mathrm{Tr}_{K_i/\mathbb{Q}}$ can be computed with at most $2\log\varphi(m_i)$ additions and $2\log\varphi(m_i)$ automorphisms.

**Algorithm 2** $\mathsf{Tr}(\mathbf{m}, m)$

**Input:** A polynomial $\mathbf{m} \in \mathbb{Q}(\xi_m)$ and integer $m$.
**Output:** A polynomial $\mathrm{Tr}_{\mathbb{Q}(\xi_m)/\mathbb{Q}}(\mathbf{m})$.

1: Suppose that $m_2 = 2^{s_0} p_1^{s_1} \cdots p_l^{s_l}$, where $p_i$ are odd prime numbers and pairwise coprime. Let $F_0 = \mathbb{Q}(\xi_{2^{s_0}})$, $F_i = \mathbb{Q}(\xi_{2^{s_0}}, \xi_{p_1^{s_1}}, \cdots, \xi_{p_l^{s_l}})$, for $i = 1, \ldots, l$.
2: **if** $s_0 \geq 3$ **then**
3: Suppose that $\mathrm{Gal}(F_0/\mathbb{Q}) = \langle\sigma\rangle \times \langle\tau\rangle$;
4: $\mathbf{m} = \mathsf{CircTr}(\mathbf{m}, \sigma, 2^{s_0-2})$;
5: $\mathbf{m} = \mathbf{m} + \tau(\mathbf{m})$;
6: **end if**
7: **if** $s_0 = 2$ **then**
8: $\mathbf{m} = \mathbf{m} + \sigma(\mathbf{m})$, where $\langle\sigma\rangle = \mathrm{Gal}(F_0/\mathbb{Q})$;
9: **end if**
10: **for** $i = 1$ to $l$ **do**
11: $\mathbf{m} = \mathsf{CircTr}(\mathbf{m}, \sigma_i, \varphi(p_i^{s_i}))$, where $\langle\sigma_i\rangle = \mathrm{Gal}(F_i/F_{i-1})$;
12: **end for**
13: **return** $\mathbf{m}$;

### 3.4 Matrix multiplication for plaintexts

Based on Theorem 2, for two polynomials $\mathbf{m}_A^{(\mathbf{uv})}, \mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})} \in \mathcal{R}$ encoding two matrices $A \in \mathbb{Z}^{r\times s}$ and $B \in \mathbb{Z}^{s\times t}$ with $r \leq \varphi(m_1)$, $s \leq \varphi(m_2)$ and $t \leq \varphi(m_3)$ respectively, we can use one polynomial multiplication and one trace operation to obtain a new polynomial that encodes $A \cdot B$. Based on Proposition 2, the trace operation in lines 2 requires at most $2\log\varphi(m_2)$ automorphisms. Therefore, the algorithm requires at most $2\log\varphi(m_2)$ automorphisms and one polynomial multiplication. There observations yield the following Algorithm 3.

**Algorithm 3** Matrix multiplication for plaintexts

**Input:** $\mathbf{m}_A^{(\mathbf{uv})}, \mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})} \in \mathcal{R}$ that encode matrices $A \in \mathbb{Z}^{r\times s}$ and $B \in \mathbb{Z}^{s\times t}$.
**Output:** A polynomial $\mathbf{m}_{AB}^{(\mathbf{uw})} \in \mathcal{R}$ encoding $A \cdot B$.

1: $\mathbf{m}_{AB}^{(\mathbf{uw})} = \mathrm{Tr}_{K/K_{13}}(\mathbf{m}_A^{(\mathbf{uv})} \cdot \mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})})$;
2: **return** $\mathbf{m}_{AB}^{(\mathbf{uw})}$;

***Remark 2*** In the general setting of $r, s, t$, let $l_1 = \lceil \frac{r}{\varphi(m_1)} \rceil$, $l_2 = \lceil \frac{s}{\varphi(m_2)} \rceil$, and $l_3 = \lceil \frac{t}{\varphi(m_3)} \rceil$. Then, the matrices $A \in \mathbb{Z}^{r\times s}$ and $B \in \mathbb{Z}^{s\times t}$ are partitioned into $l_1 \times l_2$ and $l_2 \times l_3$ submatrices, respectively. Consequently, the block matrix multiplication requires $l_1 l_2 l_3$ polynomial multiplications and at most $2l_1 l_2 l_3 \log(\varphi(m_2))$ automorphisms. By employing a technique from [32], the number of required automorphisms can be reduced to $2l_1 l_3 \log(\varphi(m_2))$.

## 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.

### 4.1 Homomorphic trace computation

Let $K = \mathbb{Q}(\xi_m)$ and $F = \mathbb{Q}(\xi_{m_1 m_3})$. Firstly, we consider the case when $\mathrm{Gal}(K/F)$ is cyclic. Algorithm 1 can be extended directly to the homomorphic case, see Algorithm 4 below, where $\sigma$ is a generator of $\mathrm{Gal}(K/F)$, and $r = |\mathrm{Gal}(K/F)|$.

**Proposition 3** *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 $E$. If $\mathrm{Gal}(K/F)$ is cyclic, then by Algorithm 4, $\mathsf{HomCircTr}(\mathbf{c}, \sigma, r)$ 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 $B_{\mathrm{ks}}$ is given in (6).*

***Proof*** The correctness of Algorithm 4 is based on the correctness of Algorithm 1 and homomorphic automorphism. The proof of the noise bound is given in Appendix A. □

**Algorithm 4** $\mathsf{HomCircTr}(\mathbf{c}, \sigma, r)$

**Input**:

1. A ciphertext $\mathbf{c} \in \mathcal{R}_Q^2$ encrypting $\mathbf{m} \in \mathcal{R}_p$.
2. An automorphism $\sigma$ and an integer $r$.

**Output:** A ciphertext $\mathbf{c}_r$ encrypting $\sum_{i=0}^{r-1} \sigma^i(\mathbf{m}) \in \mathcal{R}_p$.

**if** $r = 1$ **then**
  **return** $\mathbf{c}$;
**end if**
**if** $r$ is even **then**
  $\mathbf{c}_{r/2} = \mathsf{HomCircTr}(\mathbf{c}, \sigma, r/2)$;
  **return** $\mathbf{c}_{r/2} + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^{r/2}}, \mathbf{c}_{r/2})$;
**else**
  $\mathbf{c}_{(r-1)/2} = \mathsf{HomCircTr}(\mathbf{c}, \sigma, (r-1)/2)$;
  **return** $\mathbf{c}_{(r-1)/2} + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^{\frac{r-1}{2}}}, \mathbf{c}_{\frac{r-1}{2}}) + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^r}, \mathbf{c})$;
**end if**

For an arbitrary cyclotomic field extension, based on Proposition 2 and Algorithm 2, we can present an algorithm to compute the trace operator in the homomorphic case. In Algorithm 5, we use the fact that $\mathrm{Gal}(K/K_{13}) \simeq \mathrm{Gal}(\mathbb{Q}(\xi_{m_2})/\mathbb{Q})$.

**Proposition 4** *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 by Algorithm 5, $\mathsf{HomTr}_{K/K_{13}}(\mathbf{c})$ outputs a ciphertext $\mathbf{c}$ encrypting $\mathrm{Tr}_{K/K_{13}}(\mathbf{m})$ with noise bounded by $\varphi(m_2)E + (\varphi(m_2)-1)B_{\mathrm{ks}}$, where $B_{\mathrm{ks}}$ is given in (6).*

***Proof*** The correctness of Algorithm 5 is based on the correctness of Algorithm 2. The proof of the noise bound is given in Appendix A. □

### 4.2 Homomorphic matrix multiplication

Since polynomial additions, polynomial multiplications and trace functions in Algorithm 3 can be executed homomorphically, the framework proposed in Sect. 3 can be extended to support homomorphic matrix multiplication.

We present a new framework for homomorphic matrix multiplication below. It mainly consists of three subprograms: Encoding and Encryption (Algorithm 6), Decryption and Decoding (Algorithm 8), and Homomorphic Matrix Multiplication (Algorithm 7).

**Algorithm 5** $\mathsf{HomTr}_{K/K_{13}}(\mathbf{c})$

**Input**: A ciphertext $\mathbf{c} \in \mathcal{R}_Q^2$ encrypting $\mathbf{m} \in \mathcal{R}_p$;

**Output:** A ciphertext $\mathbf{c}$ encrypting $\mathrm{Tr}_{K/K_{13}}(\mathbf{m})$.

Suppose that $m_2 = 2^{s_0} p_1^{s_1} \cdots p_l^{s_l}$, where $p_i$ are odd prime numbers and pairwise coprime. Let $F_0 = K_{13}(\xi_{2^{s_0}})$, $F_i = K_{13}(\xi_{2^{s_0}}, \xi_{p_1^{s_1}}, \cdots, \xi_{p_l^{s_l}})$, for $i = 1, \ldots, l$.
**if** $s_0 \geq 3$ **then**
  Suppose that $\mathrm{Gal}(F_0/K_{13}) = \langle\sigma\rangle \times \langle\tau\rangle$;
  $\mathbf{c} = \mathsf{HomCircTr}(\mathbf{c}, \sigma, 2^{s_0-2})$;
  $\mathbf{c} = \mathbf{c} + \mathsf{BGV.Auto}(\mathsf{atk}_\tau, \mathbf{c})$;
**end if**
**if** $s_0 = 2$ **then**
  $\mathbf{c} = \mathbf{c} + \mathsf{BGV.Auto}(\mathsf{atk}_\sigma, \mathbf{c})$, where $\langle\sigma\rangle = \mathrm{Gal}(F_0/K_{13})$;
**end if**
**for** $i = 1$ to $l$ **do**
  $\mathbf{c} = \mathsf{HomCircTr}(\mathbf{c}, \sigma_i, \varphi(p_i^{s_i}))$, where $\langle\sigma_i\rangle = \mathrm{Gal}(F_i/F_{i-1})$;
**end for**
**return** $\mathbf{c}$;

***Remark 3*** For $\mathbf{v}_i^\vee \in \mathbb{Q}[x]/(\Phi_m(x))$, suppose that $a$ is the lowest common multiple of the denominators of coefficients of $\mathbf{v}_i^\vee$. We can choose the plaintext modulus $p$ such that $a$ is invertible in $\mathbb{Z}_p$. Then $\mathbf{v}_i^\vee$ can be regarded as an element in $\mathbb{Z}_p[x]/(\Phi_m(x))$ by setting $\mathbf{v}_i^\vee := \mathbf{v}_i^\vee \cdot a \cdot b \mod p$, where $b$ is the inverse of $a$ in $\mathbb{Z}_p$.

**Algorithm 6** Encoding and encryption

**Input:** A matrix $A = (a_{ij}) \in \mathbb{Z}_p^{r\times s}$ or $\mathbb{Z}_p^{s\times t}$, the public key $\mathsf{pk}$, and an integer $b \in \{0, 1\}$;

**Output:** $\mathbf{c}_A$, a ciphertext encrypting $\mathbf{m}_A^{(\mathbf{uv})}$ or $\mathbf{m}_A^{(\mathbf{v}^\vee\mathbf{w})}$ which encodes $A$.
**if** $b = 1$ **then**
  $\mathbf{m} = \sum_{i=1}^{r}\sum_{j=1}^{s} a_{ij}\mathbf{u}_i\mathbf{v}_j$;
**else**
  $\mathbf{m} = \sum_{i=1}^{s}\sum_{j=1}^{t} a_{ij}\mathbf{v}_i^\vee\mathbf{w}_j$;
**end if**
$\mathbf{c}_A = \mathsf{BGV.Enc}(\mathsf{pk}, \mathbf{m})$;
**return** $\mathbf{c}_A$;

**Algorithm 7** Homomorphic matrix multiplication

**Input:**

1. $\mathbf{c}_A$, $\mathbf{c}_B$ are two BGV ciphertexts encrypting $\mathbf{m}_A^{(\mathbf{uv})}$ and $\mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})}$ which encode $A \in \mathbb{Z}_p^{r\times s}$ and $B \in \mathbb{Z}_p^{s\times t}$, respectively;
2. Automorphism keys for homomorphic trace computing;
3. Relinearization key $\mathsf{rlk}$.

**Output:** $\mathbf{c}$, where $\mathbf{c}$ is a ciphertext encrypting $\mathbf{m}_{AB}^{(\mathbf{uw})}$ which encodes $AB \in \mathbb{Z}_p^{r\times t}$.
$\mathbf{c} = \mathsf{BGV.Mult}(\mathsf{rlk}, \mathbf{c}_1, \mathbf{c}_2)$;
$\mathbf{c} = \mathsf{HomTr}_{K/K_{13}}(\mathbf{c})$;
**return** $\mathbf{c}$;

**Algorithm 8** Decryption and decoding

**Input:**

1. $\mathbf{c}_C$, a ciphertext encrypting $\mathbf{m}_C^{(\mathbf{uw})}$ which encodes $C = (c_{ij}) \in \mathbb{Z}_p^{r\times t}$;
2. secret key $\mathbf{s}$.

**Output:** Matrix $C \in \mathbb{Z}_p^{r\times t}$.

1: $\mathbf{m}_C = \mathsf{BGV.Dec}(\mathbf{s}, \mathbf{c}_C)$;
2: Rewrite $\mathbf{m}_C$ as $\sum_{i=1}^{r}\sum_{j=1}^{t} c_{ij}\mathbf{u}_i\mathbf{w}_j$.
3: **return** $(c_{ij} \mod p)$;

### 4.2.1 Noise estimation

Without loss of generality, suppose that the input ciphertexts of Algorithm 7 have the same modulus $Q$.

**Proposition 5** *Let $\mathbf{c}_A, \mathbf{c}_B \in \mathcal{R}_Q^2$ be two BGV ciphertexts encrypting $\mathbf{m}_A^{(\mathbf{uv})}$ and $\mathbf{m}_B^{(\mathbf{v}^\vee\mathbf{w})}$ with noise $\mathbf{e}_1$ and $\mathbf{e}_2$, respectively. Suppose that $\|\mathbf{e}_1\|_{\mathrm{can}} \le E_1$ and $\|\mathbf{e}_2\|_{\mathrm{can}} \le E_2$. Then Algorithm 7 outputs a ciphertext $\mathbf{c} \in \mathcal{R}_q^2$ encrypting $\mathbf{m}_{AB}^{(\mathbf{uw})}$ with noise bounded by*

$$\begin{aligned} E &= \varphi(m_2)\left(\left(\frac{q}{Q}E_1 + B_{\mathrm{scale}}\right)\left(\frac{q}{Q}E_2 + B_{\mathrm{scale}}\right) + B_{\mathrm{ks}}\right) + (\varphi(m_2)-1)B_{\mathrm{ks}} \\ &= O(hp^2n^2), \end{aligned}$$

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

The proof of Proposition 5 is given in Appendix A.

### 4.2.2 Complexity analysis

In Algorithm 7, line 2 requires one ciphertext-ciphertext multiplication and line 3 require at most $2\log s$ homomorphic automorphisms. Overall, Algorithm 7 requires one ciphertext-ciphertext multiplication and at most $2\log s$ homomorphic automorphisms.

## 5 Implementation and comparison with previous work

Our implementation of the new framework and its algorithms is publicly available at https://github.com/XiaopengZheng/HEMat_nonsquare

We provide more detailed comparisons under specific parameters in Table 2.

1. For convenience in implementation, to securely multiply two matrices $A \in \mathbb{Z}_p^{r\times s}$ and $B \in \mathbb{Z}_p^{s\times t}$, we choose $m = p_1p_2p_3$, where $p_1$, $p_2$, $p_3$ are all prime numbers, such that $\varphi(p_1) \approx r$, $\varphi(p_2) \approx s$ and $\varphi(p_3) \approx t$, and use zero padding if $rst < n$.
2. The method in [17] only supports the case when $A \in \mathbb{Z}_p^{r\times s}$ and $B \in \mathbb{Z}_p^{s\times t}$ with $s = t$ and the method in [16] only supports the cases when $s = t$ or $r = t$.

3. For the methods in [4], we choose $p_1, p_2, p_3$ to be as small as possible such that they are pairwise coprime and satisfy $p_1 \geq r$, $p_2 \geq s$, and $p_3 \geq t$. Furthermore, for the method BMM-I in [4], we choose the dimension of ring $n$ to be power of two and as small as possible such that $n/4 \geq \max\{p_1p_2, p_1p_3, p_2p_3\}$. For the method BMM-II in [4], we choose the dimension of ring $n$ to be power of two and as small as possible such that $n/2 \geq p_1p_2p_3$.
4. The level of security is $\lambda \geq 80$ bits. The security level is estimated by the program in HElib [28].
5. The cyphertext modulus $Q$ is chosen such that the scheme supports one matrix multiplication. Since the the multiplication depth in the new method in only one ciphertext-ciphertext multiplication, we can choose a smaller $Q$.
6. The platform is a personal laptop with AMD Ryzen 7 6800 H, Radeon Graphics of 64GB Memory running, Ubuntu 22.04.2 LTS and only one thread is used.

**Comparison With Most-Recent Algorithms.** In a recent work [4], Chen et al. presented a bicyclic encoding technique for matrix encoding. For the computation of the multiplication of two matrices $A \in \mathbb{Z}^{r\times s}$ and $B \in \mathbb{Z}^{s\times t}$, compared with the BMM-II algorithm in [4], our algorithm performs slightly better than theirs, particularly when $s$ is small. Specifically, the BMM-II method in [4] requires $\lceil \log r \rceil + \lceil \log s \rceil + \lceil \log t \rceil$ automorphisms for one matrix multiplication, while our method requires approximately $\log s$ automorphisms. For example, their algorithm costs 1 multiplication and 14 automorphisms for a $(50, 4, 50)$ matrix multiplication, while our method requires only 1 multiplication and 2 automorphisms.

In a recent work [11], Gao et al. presented a homomorphic encryption scheme called GMS for encrypted matrix multiplication. The source code for this work does not appear to be publicly available. Therefore, we have used the reported execution times from their papers. For an encrypted matrix multiplication of size $(16, 128, 4)$, the algorithm in [11] takes about 25 s on a 2.6 GHz CPU, while our algorithm requires only 0.1606 s. Similar observations hold for other rectangular cases as well. However, the algorithm in [11] based on a new homomorphic encryption scheme called GMS, so such significant performance differences may be attributed to the use of different optimization techniques, implementation details, or parameter settings. Therefore, it seems difficult to make a fair comparison between their algorithm and ours.

## 6 Conclusion

The paper presents a new framework for homomorphic matrix multiplication. It is shown to be more efficient compared with all existing methods, in that it requires fewer basic homomorphic operations. Experimental results show that the new framework has the best performance for matrix multiplication when $r \cdot s \cdot t \leq n$. It is 1.2 to 106.8 times faster than other existing algorithms.

## A Appendix: noise estimates

**Proposition** 3. 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 $E$. If $\mathrm{Gal}(K/F)$ is cyclic, then by Algorithm 4, $\mathsf{HomCircTr}_{K/F}(\mathbf{c}, r)$ 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 $B_{\mathrm{ks}}$ is given in (6).

***Proof*** When $\ell = 1$, then Algorithm 4 outputs $\mathbf{c}$ directly, the conclusion holds trivially. Assume that the conclusion holds for $\ell < k$. For $\ell = k$, if $\ell$ is even, then

$$\mathbf{c}_\ell = \mathbf{c}_{\ell/2} + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^{\ell/2}}, \mathbf{c}_{\ell/2}). \tag{14}$$

By induction 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 (14), the noise $\mathbf{e}_\ell$ of $\mathbf{c}_l$ satisfies

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

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

If $\ell$ is odd, then

$$\begin{aligned}\mathbf{c}_\ell = \ & \mathbf{c}_{(\ell-1)/2} + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^{(\ell-1)/2}}, \mathbf{c}_{\ell/2}) \\ & + \mathsf{BGV.Auto}(\mathsf{atk}_{\sigma^\ell}, \mathbf{c})\end{aligned} \tag{15}$$

By induction 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 (15), the noise $\mathbf{e}_\ell$ of $\mathbf{c}_\ell$ satisfies

$$\begin{aligned}\|\mathbf{e}_\ell\|_{\mathrm{can}} &\le \|\mathbf{e}_{\frac{\ell-1}{2}}\|_{\mathrm{can}} + \|\mathbf{e}_{\frac{\ell-1}{2}} + \mathbf{e}_{\mathrm{ks}}\|_{\mathrm{can}} + \|\mathbf{e} + \mathbf{e}_{\mathrm{ks}}\|_{\mathrm{can}} \\ &\le \ell(E + B_{\mathrm{ks}}) - B_{\mathrm{ks}}.\end{aligned}$$

□

**Proposition 4.** 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 by Algorithm 5, $\mathsf{HomTr}_{K/K_{13}}(\mathbf{c}, m)$ outputs a ciphertext $\mathbf{c}$ encrypting $\mathrm{Tr}_{K/K_{13}}(\mathbf{m})$ with noise bounded by $\varphi(m)E + (\varphi(m) - 1)B_{\mathrm{ks}}$, where $B_{\mathrm{ks}}$ is given in (6).

***Proof*** Suppose that $m_2 = 2^{s_0} p_1^{s_1} \cdots p_l^{m_l}$, where $p_i$ are odd prime numbers and pairwise coprime. Denote $\varphi(2^{s_0})$ by $r_0$ and $\varphi(p_i^{s_i})$ by $r_i$, for $i = 1, \ldots, l$. By a same proof as Proposition 3, in Line 5 or Line 8 of Algorithm 5, the noise of $\mathbf{c}$ is bounded by $r_0 E + (r_0 - 1)B_{\mathrm{ks}}$. In Line 11, when $i = 1$, we have $\mathbf{c} := \mathsf{HomCircTr}_{F_1/F_0}(\mathbf{c}, r_1)$. Then by Proposition 3, the noise of $\mathbf{c}$ is bounded by

$$r_1(r_0 E + (r_0 - 1)B_{\mathrm{ks}}) + (r_1 - 1)B_{\mathrm{ks}} \le r_0 r_1 E + (r_0 r_1 - 1)B_{\mathrm{ks}}.$$

By induction, we obtain that the noise of the final ciphertext $\mathbf{c}$ is bounded by

$$r_0 r_1 \cdots r_l E + (r_0 r_1 \cdots r_l - 1)B_{\mathrm{ks}} \le \varphi(m_2)E + (\varphi(m_2) - 1)B_{\mathrm{ks}}.$$

□

**Proposition 5.** Let $\mathbf{c}_A, \mathbf{c}_B \in \mathcal{R}_Q^2$ be two BGV ciphertexts encrypting $\mathbf{m}_A^{(\mathbf{uv})}$ and $\mathbf{m}_B^{(\mathbf{v}^\vee \mathbf{w})}$ with noise $\mathbf{e}_1$ and $\mathbf{e}_2$, respectively. Suppose that $\|\mathbf{e}_1\|_{\mathrm{can}} \le E_1$ and $\|\mathbf{e}_2\|_{\mathrm{can}} \le E_2$. Then Algorithm 7 outputs a ciphertext $\mathbf{c} \in \mathcal{R}_q^2$ encrypting $\mathbf{m}_{AB}^{(\mathbf{uw})}$ with noise bounded by

$$\begin{aligned}E &= s\left(\left(\frac{q}{Q}E_1 + B_{\mathrm{scale}}\right)\left(\frac{q}{Q}E_2 + B_{\mathrm{scale}}\right) + B_{\mathrm{ks}}\right) + (s-1)B_{\mathrm{ks}} \\ &= O(hp^2n^2),\end{aligned}$$

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

***Proof*** By Theorem 2,

$$\mathrm{Tr}_{K/K_{13}}\left(\mathbf{m}_A^{(\mathbf{uv})} \cdot \mathbf{m}_B^{(\mathbf{v}^\vee \mathbf{w})}\right) = \mathbf{m}_C^{(\mathbf{uw})}. \tag{16}$$

Then $\mathbf{c} \in \mathcal{R}_q^2$ in line 3 of Algorithm 7 encrypts $\mathbf{m}_C^{(\mathbf{uw})}$ by the correctness of Algorithm 5. By Lemma 4, the noise after ciphertext multiplication is bounded by

$$\left(\frac{q}{Q}E_1 + B_{\mathrm{scale}}\right)\left(\frac{q}{Q}E_2 + B_{\mathrm{scale}}\right) + B_{\mathrm{sk}}.$$

where $(q/Q_\ell)E_\ell$ is approximately equal to or less than $B_{\mathrm{scale}}$, for $\ell = 1, 2$. Then by Proposition 4, after homomorphic trace operation, the noise of the ciphertext $\mathbf{c} \in \mathcal{R}_q^2$ is bounded by

$$\varphi(m_2)\left(\left(\frac{q}{Q}E_1 + B_{\mathrm{scale}}\right)\left(\frac{q}{Q}E_2 + B_{\mathrm{scale}}\right) + B_{\mathrm{ks}}\right) + (\varphi(m_2) - 1)B_{\mathrm{ks}}.$$

According to the choice of $q$, $B_{\mathrm{scale}} = p\left(\sqrt{3n} + 8\sqrt{nh/3}\right)$ and $\varphi(m_2) = O(n)$, we have

$$E = O(sB_{\mathrm{scale}}^2) = O(hp^2n^2).$$

□

**Acknowledgements** This research was supported by the National Key R&D Program of China under Grant No. 2020YFA0712300, the National Natural Science Foundation of China under Grant No. 12171469, and STU Scientific Research Initiation Grant No. NTF24023T.

**Author contributions** Xiaopeng Zheng: conceptualization, methodology, writing original draft, software. Hongbo Li: methodology, writing review and editing, funding acquisition. DingkangWang: writing review and editing, funding acquisition, validation.

**Data availability** No datasets were generated or analysed during the current study.

## Declarations

**Competing interests** The authors declare no competing interests.

## References

1. Alperin-Sheriff J., Peikert C.: Practical bootstrapping in quasilinear time. In: Annual Cryptology Conference. Springer, New York, pp. 1–20 (2013).
2. Brakerski Z., Gentry C., Vaikuntanathan V.: (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory **6**(3), 1–36 (2014).
3. Chen H., Dai W., Kim M., Song Y.: Efficient homomorphic conversion between (ring) LWE ciphertexts. In: International Conference on Applied Cryptography and Network Security, Springer, pp. 460–479 (2021).
4. Chen J., Yang L., Wu W., Liu Y., Feng Y.: Homomorphic matrix operations under bicyclic encoding. IEEE Trans. Inf. Forensics Secur. **20**, 1390–1404 (2025). https://doi.org/10.1109/TIFS.2024.3490862.
5. Cheon J.H., Kim A., Kim M., Song Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology—ASIACRYPT 2017, Springer, Cham, pp. 409–437 (2017).
6. Chillotti I., Gama N., Georgieva M., Izabachène M.: Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds. In: Advances in Cryptology—ASIACRYPT 2016, Springer, Cham, pp. 3–33 (2016).
7. 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, Springer, pp. 325–340 (2016).

8. Ducas L., Micciancio D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Advances in Cryptology—EUROCRYPT 2015, Springer, Cham, pp. 617–640 (2015).
9. Duong D.H., Mishra P.K., Yasuda M.: Efficient secure matrix multiplication over lwe-based homomorphic encryption. Tatra Mt. Math. Publ. **67**(1), 69–83 (2016).
10. Fan J., Vercauteren F.: Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Paper 2012/144 (2012).
11. Gao J., Gao Y.: Gms: an efficient fully homomorphic encryption scheme for secure outsourced matrix multiplication. J. Supercomput. **80**(18), 26435–26461 (2024).
12. Gentry C., Sahai A., Waters B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013, Springer, Cham, pp. 75–92 (2013).
13. Gentry C.: Fully homomorphic encryption using ideal lattices. In: STOC'09—Proceedings of the 2009 ACM International Symposium on Theory of Computing, pp. 169–178. ACM, New York (2009).
14. Halevi S., Shoup V.: Design and implementation of a homomorphic-encryption library. IBM Res. **6**(12–15), 8–36 (2013).
15. Huang H., Zong H.: Secure matrix multiplication based on fully homomorphic encryption. J. Supercomput. **79**(5), 5064–5085 (2023).
16. Huang Z., Hong C., Weng C., Lu W., Qu H.: More efficient secure matrix multiplication for unbalanced recommender systems. IEEE Trans. Depend. Secure Comput. **20**(01), 551–562 (2023).
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).
18. Juvekar C., Vaikuntanathan V., Chandrakasan A.: {GAZELLE}: a low latency framework for secure neural network inference. In: 27th USENIX Security Symposium (USENIX Security 18), pp. 1651–1669 (2018).
19. Lee E., Lee J.-W., Lee J., Kim Y.-S., Kim Y., No J.-S., Choi W.: Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In: International Conference on Machine Learning, PMLR, pp. 12403–12422 (2022).
20. 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: Advances in Cryptology—EUROCRYPT 2023, Springer, Cham, pp. 227–256 (2023).
21. Liu F.-H., Wang H.: Batch bootstrapping I: a new framework for SIMD bootstrapping in polynomial modulus. In: Advances in Cryptology—EUROCRYPT 2023, Springer, Cham, pp. 321–352 (2023).
22. Lu W.-J., Sakuma J.: More practical privacy-preserving machine learning as a service via efficient secure matrix multiplication. In: Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp. 25–36 (2018).
23. Lyubashevsky V., Peikert C., Regev O.: A toolkit for ring-LWE cryptography. In: Advances in Cryptology–EUROCRYPT 2013, Springer, New York, pp. 35–54 (2013).
24. Lyubashevsky V., Peikert C., Regev O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology–EUROCRYPT 2010, Springer, Cham, pp. 1–23 (2010).
25. Mishra P.K., Duong D.H., Yasuda M.: Enhancement for secure multiple matrix multiplications over ring-lwe homomorphic encryption. In: Information Security Practice and Experience: 13th International Conference, ISPEC 2017, Melbourne, VIC, Australia, December 13–15, 2017, Proceedings 13, Springer, pp. 320–330 (2017).
26. Pang Q., Zhu J., Möllering H., Zheng W., Schneider T.: Bolt: Privacy-preserving, accurate and efficient inference for transformers. In: 2024 IEEE Symposium on Security and Privacy (SP), IEEE, pp. 4753–4771 (2024).
27. Regev O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 1–40 (2009).
28. Shai H., Victor S.: Design and implementation of HElib: a homomorphic encryption library. Cryptology ePrint Archive, Paper 2020/1481 (2020).
29. Vinogradov I.M.: Elements of Number Theory. Courier Dover Publications, New York (2003).
30. Xia H., Liu F.-H., Wang H.: More efficient functional bootstrapping for general functions in polynomial modulus. In: Theory of Cryptography Conference, Springer, pp. 130–163 (2025).
31. Zhang J., Yang X., He L., Chen K., Lu W.-J., Wang Y., Hou X., Liu J., Ren K., Yang X.: Secure Transformer Inference Made Non-interactive. Cryptology ePrint Archive, Paper 2024/136 (2024). https://eprint.iacr.org/2024/136.
32. Zheng X., Li H., Wang D.: A New Framework for Fast Homomorphic Matrix Multiplication. Cryptology ePrint Archive, Paper 2023/1649 (2023). https://eprint.iacr.org/2023/1649.

33. Zhu L., Hua Q., Chen Y., Jin H.: Secure outsourced matrix multiplication with fully homomorphic encryption. In: European Symposium on Research in Computer Security, Springer, Cham, pp. 249–269 (2023).