The Journal of Supercomputing (2024) 80:26435–26461
https://doi.org/10.1007/s11227-024-06449-3

GMS: an efficient fully homomorphic encryption scheme for secure outsourced matrix multiplication

Jianxin Gao[1] · **Ying Gao**[1,2]

Accepted: 11 August 2024 / Published online: 26 August 2024

Abstract

Fully homomorphic encryption (FHE) is capable of handling sensitive encrypted data in untrusted computing environments. The efficient application of FHE schemes in secure outsourced computation can effectively address security and privacy concerns. This paper presents a novel fully homomorphic encryption scheme called GMS, based on the n-secret learning with errors (LWE) assumption. By utilizing block matrix and decomposition technology, GMS achieves shorter encryption and decryption times and smaller ciphertext sizes compared to existing FHE schemes. For secure outsourced matrix multiplication $\mathbf{A}_{m\times n} \cdot \mathbf{B}_{n\times l}$ with arbitrary dimensions, GMS only requires $O(\max\{m, n, l\})$ rotations and one homomorphic multiplication. Compared to the state-of-the-art methods, our approach stands out by achieving a significant reduction in the number of rotations by a factor of $O(\log \max\{n, l\})$, along with a decrease in the number of homomorphic multiplications by a factor of n and $O(\min\{m, n, l\})$. The experimental results demonstrate that GMS shows superior performance for secure outsourced matrix multiplication of any dimension. For example, when encrypting a 64×64-dimensional matrix, the size of the ciphertext is only 1.27 MB. The encryption and decryption process takes approximately 0.2 s. For matrix multiplication $\mathbf{A}_{64\times 64} \cdot \mathbf{B}_{64\times 64}$, the runtime of our method is 39.98 s, achieving a speedup of up to 5X and 2X.

Keywords Fully homomorphic encryption (FHE) · Secure outsourced computation · n-Secret learning with errors (n-secret LWE) · Matrix multiplication

✉ Ying Gao
gaoying@buaa.edu.cn

Jianxin Gao
2239106@buaa.edu.cn

[1] School of Cyber Science and Technology, Beihang University, No.37 Xueyuan Road, Haidian District, Beijing 100191, China

[2] Zhongguancun Laboratory, Beijing 100094, China

1 Introduction

Cloud computing technology provides clients with instant access to shared storage and computing resources, reducing investment and IT system maintenance costs. The rapid development of cloud computing has led to the widespread use of cloud outsourcing. This allows clients with limited computing resources to delegate intensive and complex computing tasks to high-performance cloud servers. However, the rapid development of cloud computing still faces several challenges:

(1) Data privacy: Data outsourced by clients to cloud servers is typically confidential, such as medical or trade secret information. It is important to ensure that this data is kept secure and protected from unauthorized access. The shared characteristics of cloud servers can potentially expose private data, making it difficult to ensure privacy during outsourced computation.
(2) Efficiency: Outsourced computation currently employs privacy protection technology to ensure the confidentiality of data. Incorporating privacy protection directly, however, can result in significant costs. Therefore, clients require an efficient process for outsourced computational tasks while still ensuring privacy.

Privacy protection technologies, such as fully homomorphic encryption (FHE), secure multi-party computing (MPC) [1], and disguise-based techniques [2], are commonly used to ensure data privacy in secure outsourced computation. It is important to note that while these technologies provide enhanced privacy, they may also impact performance. Secure multi-party computing is a general framework for such tasks, but it can result in high communication overhead and require multiple rounds of interaction between the client and the cloud server. The process of secure outsourced computation based on MPC may not be very efficient. The disguise-based technique utilizes the unique properties of matrices to create a specialized protocol for secure matrix multiplication. However, the representation of security symbols in various protocols is not standardized, which can potentially lead to information leakage. The flaw in the current disguise-based secure outsourced computation poses a significant threat to data privacy. However, achieving composability has not been possible with this technique. On the other hand, secure outsourced computation based on FHE is non-interactive and incurs a very low communication cost for uploading and downloading. Furthermore, secure outsourced computation based on FHE relies on encryption and decryption techniques from cryptography. This approach ensures data privacy, prevents information disclosure, and enables data composability. As a result, secure outsourced computation based on fully homomorphic encryption technology offers significant advantages and is widely used.

Most scholars focus on secure outsourced computation for matrix multiplication. Matrix multiplication is a widely used operation in scientific and engineering fields, such as statistical analysis, image processing, and machine learning. It can solve many practical problems. For instance, the training and prediction process of deep learning models can be simplified to a series of fundamental matrix operations. Additionally, an image can be represented and stored as a matrix based on its pixel

characteristics, and the image processing process can be transformed into a matrix operation process. Efficient and secure outsourced computation for matrix multiplication is crucial for secure computation applications. However, the current fully homomorphic encryption scheme used to secure outsourced matrix multiplication has limitations. It usually involves a packaging process [3–5] that matches the data type of the plaintext. Additionally, it is restricted to applications involving binary matrices [6]. These limitations present challenges. Therefore, designing and implementing an efficient and secure outsourced matrix multiplication based on fully homomorphic encryption technology has become an important research problem.

1.1 Related work

Currently, secure outsourced matrix multiplication based on fully homomorphic encryption is typically implemented using existing schemes. This section provides an introduction to fully homomorphic encryption techniques, as well as a discussion of the advantages and limitations of current research on secure outsourced matrix multiplication using fully homomorphic encryption.

1.1.1 Fully homomorphic encryption

Fully homomorphic encryption allows computations on the ciphertext without requiring knowledge of the private key, making it an important tool for privacy protection. Most existing fully homomorphic encryption schemes are based on the AGCD assumption [7], the LWE assumption [8] and the NTRU assumption [9]. Fully homomorphic encryption based on the AGCD assumption revolves around integer encryption. This is considered to be the first generation of fully homomorphic encryption. It is easy to understand but requires a large public key and has low efficiency. The NTRU assumption is a non-standard assumption, and many researchers have questioned its security. The work of L*ó*peze-Alt et al. [9] has received considerable criticism in recent years. Fully homomorphic encryption based on the NTRU assumption is primarily used in application scenarios with low security requirements. A significant number of existing fully homomorphic encryption schemes are based on the learning with errors (LWE) assumption, first proposed by Regev in 2009 [8]. In 2010, Regev elaborated on the hardness of the LWE problem, which can be reduced to a worst-case hard problem [10]. Since there is currently no effective quantum solution algorithm for the LWE assumption, the encryption scheme based on the LWE assumption is considered to be resistant to quantum attacks. For example, the second generation of fully homomorphic encryption BGV [11], the third generation of fully homomorphic encryption GSW [12], and the fourth generation of fully homomorphic encryption CKKS [13], among others, are all based on the LWE assumption. The following discussion primarily focuses on the fully homomorphic encryption scheme based on the LWE assumption.

The second generation of the fully homomorphic encryption scheme BGV [11] operates on the integer polynomial ring. It cleverly utilizes modulus switching and key switching to achieve ciphertext refreshing, opening up new avenues

for research in fully homomorphic encryption. Examples of other fully homomorphic encryption schemes include TFHE [14] and CKKS [13]. However, BGV has higher parameter requirements, necessitating a larger module and storage space. The fourth generation of the fully homomorphic encryption scheme CKKS [13] is a significant improvement over BGV [11]. It introduces an encoding and decoding process that allows the plaintext space to be expanded into a complex vector space. This enhancement allows the scheme to support floating-point arithmetic. The third generation of the fully homomorphic encryption scheme GSW [12] is ingenious. It exploits the properties of the eigenvalues and eigenvectors of the matrix and uses matrix decomposition to control noise expansion. As a result, the matrix decomposition technique has found widespread applications. These applications, which include schemes such as [6, 15–18], are often referred to as GSW-like schemes. These schemes primarily involve fully homomorphic encryption for matrices. HAO [6] is the first homomorphic encryption scheme for matrix that effectively controls ciphertext expansion. However, it can only encrypt plaintext for square matrices with elements in $\{0, 1\}$, which has limitations. GGH [17] is a proposed scheme based on HAO [6]. GGH [17] can encrypt square matrices without the restriction of $\{0, 1\}$, offering greater efficiency and versatility. However, since matrix multiplication does not satisfy the commutative property, the encryption process also requires the private key, essentially making it a form of symmetric encryption. It is necessary to introduce the encryption of fundamental matrices as public keys to convert it into an asymmetric encryption scheme. Subsequently, Pereira also proposed a GSW-like scheme [18], which can encrypt vectors and matrices and modify the base matrix decomposition. This scheme demonstrates relatively high efficiency. However, it still requires the introduction of n^2 base matrix encryption as public keys to achieve asymmetric encryption.

It can be seen that the existing fully homomorphic encryption schemes cannot be easily and directly applied to secure outsourced matrix multiplication. For BGV [11], TFHE [14], CKKS [13], and other fully homomorphic encryption schemes over the integer polynomial ring, the plaintext is typically represented in polynomial form. Implementing secure outsourced matrix multiplication requires packaging the matrix according to the plaintext format. There will undoubtedly be additional overhead. For GGH [17] and other fully homomorphic encryption schemes for matrices, they are inherently symmetric. This means that there are private keys involved in the encryption process that cannot be disclosed. To achieve secure matrix multiplication, it is necessary to introduce several basic matrix encryptions as public keys. However, this approach naturally leads to larger key sizes and storage requirements, which reduces efficiency. Therefore, if the fully homomorphic encryption scheme for matrices is asymmetric, then FHE-based secure outsourced matrix multiplication can eliminate the need for packaging different data types or introducing the encrypted matrix result as a public key. This can potentially lead to increased efficiency.

1.1.2 FHE-based secure outsourced matrix multiplication

Most of the existing secure outsourced matrix multiplication methods are based on fully homomorphic encryption and disguise-based technology, each of which has its own advantages. However, there are still several shortcomings in these approaches. We classify them according to the technology used. Table 1 presents the following details about the scheme: whether it supports arbitrary matrices, whether it is composable, the number of ciphertexts corresponding to a matrix (#Ciphertext), the number of homomorphic multiplications in secure outsourced matrix multiplication (#Multiplication), and the security level of the scheme. See Table 1 for more details, where n presents the dimension of matrix.

It is clear from Table 1 that the current disguise-based secure outsourced matrix multiplication generally has a low level of security. Even the only security measure that achieves CPA security [2] carries the risk of information leakage due to the use of security symbols. Performing secure outsourced matrix multiplication using fully homomorphic encryption is clearly the superior option. Then, we will introduce secure outsourced matrix multiplication based on fully homomorphic encryption in detail. In addition, we will review the existing work based on its utilization of the fully homomorphic encryption scheme.

The concept of secure outsourced matrix multiplication based on fully homomorphic encryption originates from the secure matrix–vector multiplication proposed by Halevi et al. [22]. The implementation process of secure outsourced matrix multiplication based on fully homomorphic encryption is introduced in detail in [3, 25, 26] and other literature, making it a gradually emerging research topic. The specific process of secure outsourced matrix multiplication based on fully homomorphic encryption can be described by Fig. 1. Users P1 and P2 have matrices $\mathbf{A}$ and $\mathbf{B}$,, respectively, and do not want anyone to know about them. User P3 needs to obtain

Table 1 Properties of each method

Technique	Method	Arbitrary	Composability	#Ciphertext	#Multiplication	Security
Disguise-based	Atallah et al. [19]	N	N	–	–	Non-CPA
	Lei et al. [20]	Y	N	–	–	Non-CPA
	Fu et al. [21]	N	N	–	–	Non-CPA
	Zhao et al. [2]	N	N	–	–	CPA
FHE-based	Halevi et al. [22]	N	Y	n	n	CPA
	Lu et al. [23]	Y	Y	n	n^2	CPA
	Wang et al. [24]	Y	Y	n	n^2	CPA
	Duong et al. [3]	N	N	1	1	CPA
	Lu et al. [25]	Y	N	1	1	CPA
	Hiromasa et al. [6]	N	Y	1	1	CPA
	Jiang et al. [26]	N	Y	1	n	CPA
	Huang et al. [4]	Y	Y	1	n	CPA
	Zhu et al. [5]	Y	Y	1	n	CPA

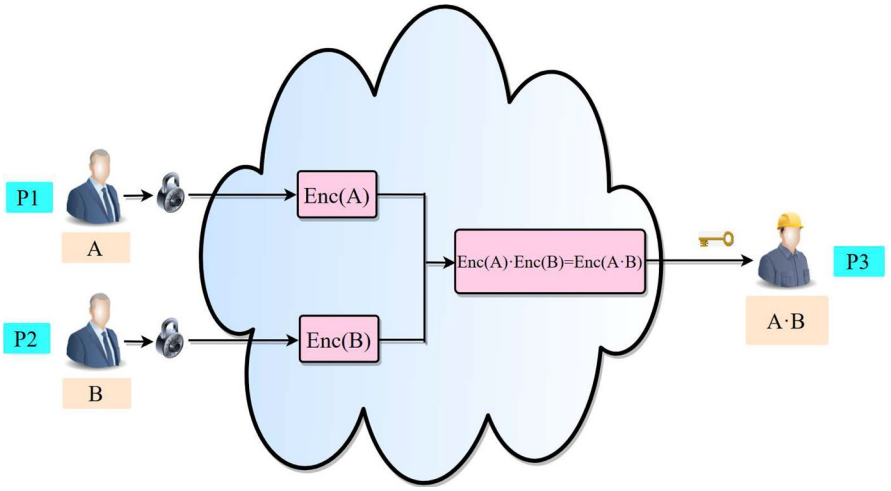

Fig. 1 FHE-based secure outsourced matrix multiplication

the product $\mathbf{A} \cdot \mathbf{B}$ without knowing the matrices $\mathbf{A}$ and $\mathbf{B}$. This can be achieved by implementing fully homomorphic encryption.

- Users P1 and P2 encrypt their respective matrices and upload the encrypted ciphertext to the cloud server.
- The cloud server performs homomorphic multiplication on the ciphertext as required and sends the result to user P3.
- User P3 receives the result and decrypts it to obtain $\mathbf{A} \cdot \mathbf{B}$.

Only [6] extended the existing fully homomorphic encryption scheme to achieve secure matrix multiplication. Hirmasa et al. [6] developed a GSW-like homomorphic encryption scheme for matrices. In this scheme, each matrix only requires one ciphertext to represent it, and each secure matrix multiplication corresponds to a homomorphic multiplication. The computational and storage overhead of this method is smaller, but it only applies to the case of a binary square matrix, specifically $\mathbf{A}_{n\times n} \cdot \mathbf{B}_{n\times n}$, where each component is either 0 or 1. There are limits to its practical application.

Halevi et al. [22] used the BGV fully homomorphic encryption to introduce the first secure matrix–vector multiplication, represented as $\boldsymbol{w} = \mathbf{A}_{n\times n} \cdot \boldsymbol{v}$, where $\boldsymbol{v}$ and $\boldsymbol{w}$ are vectors, and $\mathbf{A}_{n\times n}$ is a matrix. However, its efficiency is not high because each matrix requires n ciphertexts to represent it, and a secure matrix multiplication requires n^2 homomorphic multiplication operations. Lu et al. [23] and Wang et al. [24] used the BGV scheme to extend secure matrix–vector multiplication to secure matrix–matrix multiplication based on row order and column order, respectively. However, a matrix still requires n ciphertexts to represent it, and a secure matrix multiplication corresponds to n^2 homomorphic multiplication operations; therefore, the efficiency is still not high. Duong et al. [3] and Lu et al. [25] combined

matrix coding with the BGV scheme. In this approach, only one ciphertext is needed to represent a matrix, which reduces the storage cost. However, a disadvantage is that the result of secure multiplication does not match the encrypted format of the matrix. Therefore, the result obtained can only be decoded and output; it cannot be used for the next multiplication. This limitation hinders its practicality in terms of composability. Jiang et al. [26] combined the coding technique with CKKS/BGV fully homomorphic encryption, enabling each matrix to be represented by a single ciphertext. A secure matrix multiplication corresponds to only one homomorphic multiplication operation. The square matrix multiplication can be extended to a special form of matrix multiplication, namely $\mathbf{A}_{m\times n} \cdot \mathbf{B}_{n\times n}$, where $m \leq n$. Huang et al. [4] and Zhu et al. [5] are new studies that propose a secure outsourced matrix multiplication scheme for rectangular matrices, i.e., $\mathbf{A}_{m\times n} \cdot \mathbf{B}_{n\times l}$, which has a wider range of applications. Huang's work [4] can be accomplished by utilizing the CKKS/BGV scheme. Zhu et al. [5] have made further improvements building on the work of Huang et al. [4], which reduces the number of rotation operations and homomorphic multiplications using a novel approach. Their research aims to minimize matrix multiplication time by using the BGV scheme supported by the HElib library, which is distinguished by achieving a significant reduction in the number of rotations by a factor of $O(\log \max\{n, l\})$, as well as a decrease in the number of homomorphic multiplications by a factor of $O(n/\min\{m, n, l\})$. However, the method still requires n homomorphic multiplications for each secure matrix multiplication, and the matrix must be converted into polynomials through encoding. This introduces additional overhead.

To summarize, supporting BGV, CKKS, and other fully homomorphic encryption schemes for polynomials requires encoding and packaging, which inevitably leads to additional overhead. Additionally, there is no efficient fully homomorphic encryption scheme directly available for arbitrary matrices. Therefore, designing an efficient fully homomorphic encryption scheme for arbitrary matrices can reduce the necessity for extra processing steps, like encoding and packaging. This would simplify the representation of a matrix with only one ciphertext, enable secure outsourced matrix multiplication using just one homomorphic multiplication, and ensure that the ciphertext possesses desirable properties, such as composability. Our focus is on designing efficient fully homomorphic encryption for arbitrary matrices. These improvements will undoubtedly enhance the efficiency of secure outsourced matrix multiplication based on this scheme.

1.2 Technical details and our contribution

We start by explaining the symbols and their meanings. We denote matrices using bold capital letters. For a positive integer q, the set $\mathbb{Z}_q$ is defined as $\{0, 1, \ldots, q-1\}$. It is evident that $\mathbb{Z}_q$ forms a ring under addition and multiplication modulo q. The integer operation $\lfloor z \rceil$ is used to find the integer closest to z that satisfies $\lfloor z \rceil \in \left(z - \frac{1}{2}, z + \frac{1}{2}\right]$. The specific formula for the modular operation is

$z \bmod q = z - q\left\lfloor \frac{z}{q} \right\rfloor$. For a matrix $\mathbf{Y} = (y_{ij})$, where y_{ij} is the component in the ith row and jth column of the matrix $\mathbf{Y}$, the modulo operation can be expressed as $\mathbf{Y} \bmod q = (y_{ij} \bmod q)$, which calculates the modulus of each component of the matrix. The general overview and core techniques of our scheme are shown in Fig. 2.

We construct a fully homomorphic encryption scheme for matrices based on the n-secret LWE assumption and name it GMS. The n-secret LWE assumption can be expressed as $\mathbf{B} = (\mathbf{AS}_1 + \mathbf{E}) \bmod q$. Here, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{B} \in \mathbb{Z}_q^{n\times n}$, and $\mathbf{E} \in \mathbb{Z}_q^{n\times n}$. Each component of $\mathbf{E}$ is sampled from a discrete Gaussian distribution (error distribution) χ. To simplify the notation, we introduce several block matrices: $\mathbf{D} = (\alpha\mathbf{B}|\mathbf{A}) \in \mathbb{Z}_q^{n\times(n+m)}$, $\mathbf{V} = (\mathbf{I}_n|\mathbf{O}_{n\times m}) \in \mathbb{Z}_q^{n\times(n+m)}$, $\mathbf{S} = \begin{pmatrix} \beta\mathbf{I}_n \\ -t\mathbf{S}_1 \end{pmatrix} \in \mathbb{Z}_q^{(n+m)\times n}$, and $\mathbf{T} = \begin{pmatrix} \mathbf{I}_n \\ -\mathbf{T}_1 \end{pmatrix} \in \mathbb{Z}_q^{(n+m)\times n}$, where α, β and $t = \alpha\beta$ are all small integers. Note that $\mathbf{S}_1$ and $\mathbf{T}_1$ are smaller matrices with a reduced matrix norm. We utilize the non-unique inverse property of the block matrix $\mathbf{V}$ to illustrate the asymmetry of the scheme. This implies that disclosing the matrix $\mathbf{V}$ does not jeopardize the security of the scheme. The matrix decomposition function $G^{-1}(\cdot)$, which follows a sub-Gaussian distribution, is introduced in the scheme to reduce the matrix norm and effectively control the increase of noise. The encryption process is defined as $\mathbf{C} = (\mathbf{GTMV} + \mathbf{RD}) \bmod q$, where the gadget matrix $\mathbf{G}$ is commonly used for matrix decomposition, such as $\mathbf{G} = \boldsymbol{g} \otimes \mathbf{I}_{n+m}$. The decryption process is defined as $\mathbf{M} = (\mathbf{G}^{-1}(\eta\mathbf{V})\mathbf{CS} \bmod q) \bmod t$, which involves a private key $\mathbf{S}$ that satisfies $\mathbf{DS} = t\mathbf{E}$ and $\mathbf{VS} = \beta\mathbf{I}_n$, simplifying the decryption process. Matrix decomposition is also introduced in homomorphic operations to reduce noise amplification.

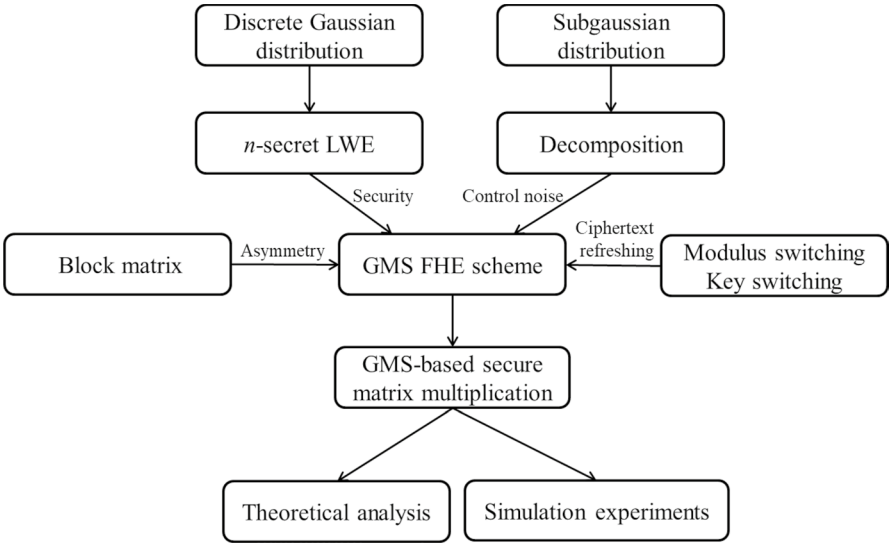

Fig. 2 Technical overview diagram

Homomorphic addition is defined as $\mathbf{C}_{add} = (\mathbf{C}_1 + \mathbf{C}_2) \bmod q$, and homomorphic multiplication is defined as $\mathbf{C}_{mul} = G^{-1}(\mathbf{C}_1)\mathbf{C}_2 \bmod q$. Additionally, for homomorphic multiplication, the rotation operation of ciphertext matrices is introduced to further enhance the efficiency of homomorphic multiplication. In addition, modulus switching and key switching techniques are defined to facilitate the process of refreshing the ciphertext and the increase in noise can be effectively controlled.

Due to the block matrix and decomposition technology utilized in the scheme design, GMS can achieve a smaller ciphertext size and lower noise expansion, resulting in shorter encryption, decryption, and evaluation times compared to existing FHE schemes. GMS can be extended directly to achieve secure outsourced matrix multiplication for arbitrary matrices by populating the matrix. Since GMS is an asymmetric encryption method for matrices, secure matrix multiplication using GMS can be implemented without the need for additional packaging or encoding. This simplifies the process and enhances efficiency.

On this basis, we are considering implementing secure outsourced matrix multiplication using GMS. We will analyze the efficiency and performance of the scheme from two perspectives: theoretical analysis and simulation experiments. The theoretical analysis primarily compared the time complexity of our scheme with existing works, while the simulation experiment tested the efficiency and performance of the scheme using C++. Specifically, we used the HElib library [22] in C++ to implement the scheme. We then ran the program on a personal laptop equipped with an 11th generation Intel(R) Core(TM) i5-11260 H CPU running at a frequency of 2.60GHz. Encryption and decryption time, ciphertext size, and homomorphic multiplication time were tested. In addition, we implemented the security outsourced matrix multiplication using GMS and replicated the last two tasks [4, 5], analyzing the time taken for matrix multiplication. Compared to existing methods [4, 5], our scheme reduces the number of homomorphic multiplications required for a matrix multiplication from n to 1. We extend the square matrix multiplication to include rectangular matrix multiplication. From the perspective of time complexity, for the matrix multiplication $\mathbf{A}_{m\times n} \cdot \mathbf{B}_{n\times l}$ of arbitrary dimensions, our approach only requires $O(\max\{m, n, l\})$ rotations, whereas [4] requires $O(n \log \max\{n, l\})$ rotations. The experimental results also demonstrate the superiority of our method.

2 Preliminaries

All matrices in this paper are defined over $\mathbb{Z}_q$, and the symbol $\|\cdot\|$ is consistently used to represent the maximum norm of a matrix. The norm $\|\mathbf{Y}\|$ refers to the maximum norm of the matrix $\mathbf{Y}$, which is defined as $\|\mathbf{Y}\| = \max_{i,j}\{|y_{i,j}|\}$. Here, y_{ij} represents the component of the matrix $\mathbf{Y}$ at the ith row and jth column position. The addition norm and multiplication norm of the matrices $\mathbf{Y}$ and $\mathbf{W}$ satisfy the following inequalities: $\|\mathbf{Y} + \mathbf{W}\| \leq \|\mathbf{Y}\| + \|\mathbf{W}\|$ and $\|\mathbf{Y} \cdot \mathbf{W}\| \leq n\|\mathbf{Y}\| \cdot \|\mathbf{W}\|$.

Here, n represents the number of columns in matrix $\mathbf{Y}$, which is also equal to the number of rows in matrix $\mathbf{W}$.

2.1 Distributions

This section focuses on the distributions required in our scheme: discrete Gaussian distributions and sub-Gaussian distributions. The error distribution in the LWE assumption follows a discrete Gaussian distribution, while the matrix decomposition follows a sub-Gaussian distribution. We will also cover gadget matrix and matrix decomposition in detail.

2.1.1 Discrete Gaussian distribution

The discrete Gaussian distribution on $\mathbb{Z}$ is introduced. This distribution is used to sample the private key in our scheme.

Definition 2.1 [27] Let c be the center and σ be the parameter of the Gaussian function over $\mathbb{Z}$, which is defined as $\rho_{\sigma,c}(x) = e^{-\frac{|x-c|^2}{2\sigma^2}}$, where $\sigma, c > 0$. Additionally, let $\rho_{\sigma,c}(\mathbb{Z}) = \sum_{i=-\infty}^{\infty} \rho_{\sigma,c}(i)$ be the discrete integral over $\mathbb{Z}$. Define $\mathcal{D}_{\sigma}(X = x) = \frac{\rho_{\sigma,c}(x)}{\rho_{\sigma,c}(\mathbb{Z})}$ as a discrete Gaussian distribution with the expectation for c and the standard deviation of σ.

If $c = 0$, we denote this special distribution as χ. We use the notation $x \leftarrow \chi$ to indicate drawing x from the distribution χ. Similarly, for a set X we use the notation $x \leftarrow X$ to indicate drawing x uniformly at random from X.

2.1.2 Sub-Gaussian distribution

The sub-Gaussian distribution is introduced. This distribution is used for matrix decomposition.

Definition 2.2 [28] A variable $X \in \mathbb{R}$ is considered sub-Gaussian, for all $t \in \mathbb{R}$, it satisfies $\mathrm{E}\left[\exp(2\pi tX)\right] \leq \exp\left(\pi s^2 t^2\right)$. In this case, X follows a sub-Gaussian distribution with parameter s.

Sub-Gaussian random variables have the following two properties that can be easily obtained from the definition of sub-Gaussian random variables:

- Homogeneity: If the sub-Gaussian random variable X has parameter s, then cX is sub-Gaussian with parameter cs.
- Pythagorean additivity: For two sub-Gaussian random variables X_1 and X_2 (that is independent from X_1) with parameter s_1 and s_2, respectively, $X_1 + X_2$ is sub-Gaussian with parameter $\sqrt{s_1^2 + s_2^2}$.

The above statement can be expanded to include vectors. A random vector $\boldsymbol{x}$ is sub-Gaussian with parameter s, if for all real unit vectors $\boldsymbol{u}$, their marginal $\langle \boldsymbol{u}, \boldsymbol{x} \rangle$ is sub-Gaussian with parameter s. It is evident from the definition that the concatenation of sub-Gaussian variables or vectors, each with a parameter s and independent of the previous one, is also sub-Gaussian with parameter s. The homogeneity and Pythagorean additivity also hold due to the linearity of vectors. It is known that the Euclidean norm of a sub-Gaussian random vector has the following upper bound.

Lemma 2.1 [28] *Let $\boldsymbol{x} \in \mathbb{R}^n$ be a random vector that has independent sub-Gaussian coordinates with parameter s. Then there exists a universal constant C such that* $\Pr\left[\|\boldsymbol{x}\| > C \cdot s\sqrt{n}\right] \leq 2^{-\Omega(n)}$.

Next, we introduce the matrix decomposition that follows the sub-Gaussian distribution. This technique is widely used in fully homomorphic encryption and has a positive impact on controlling noise growth. Let q be a positive integer, $q \geq b \geq 2$. Set $a = \lceil \log_b q \rceil$ and define a column vector $\boldsymbol{g} = \left(1, b, b^2, \ldots, b^{a-1}\right)^T$. Define the gadget matrix $\mathbf{G} = \mathbf{I}_m \otimes \boldsymbol{g} \in \mathbb{Z}^{M \times m}$, where $\mathbf{I}_m$ is the $m \times m$ dimensional identity matrix, $M = m \cdot a$, $\otimes$ denotes the tensor product. $G^{-1}(\cdot)$ represents the basis b decomposition of each component of the element $\cdot$, and is a flattening process applied to the matrix. So, for a matrix $\mathbf{Y}$, its decomposition is unique, and $G^{-1}(\mathbf{Y}) \cdot \mathbf{G} = \mathbf{Y}$, with the condition that $\|\mathbf{Y}\|$. Meanwhile, [28] has shown that the matrix decomposition function $G^{-1}(\cdot)$ follows a sub-Gaussian distribution with arguments $O(1)$.

Lemma 2.2 [28] *There is a randomized, efficiently computable function $G^{-1} : \mathbb{Z}_q^n \to \mathbb{Z}^{n \cdot \lceil \log q \rceil}$ such that for any $\boldsymbol{a} \in \mathbb{Z}_q^n$, $\boldsymbol{x} \leftarrow G^{-1}(\boldsymbol{a})$ is sub-Gaussian with parameter $O(1)$ and $\boldsymbol{a} = \mathbf{G}\boldsymbol{x} \bmod q$.*

2.2 Learning with errors

The learning with errors (LWE) assumption, originally proposed by Regev [8], has two common versions: the search version and the decision version. These versions are mutually reducible and equally secure. The security analysis of our scheme includes the decision LWE (DLWE) assumption, and the specific definitions of DLWE and security reduction are described below.

Definition 2.3 (DLWE) [8] For a security parameter λ, let $n = n(\lambda)$ be an integer dimension, let $q = q(\lambda) \geq 2$ be an integer modulus, and let $\chi = \chi(\lambda)$ be an error distribution over $\mathbb{Z}$. $\mathcal{DLWE}_{n,q,\chi}$ is the problem to distinguish the following two distributions: In the first distribution, a tuple $\left(\boldsymbol{a}_i, b_i\right)$ is sampled from uniform over $\mathbb{Z}_q^n \times \mathbb{Z}_q$; In the second distribution, $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and then a tuple $\left(\boldsymbol{a}_i, b_i\right)$ is sampled by sampling

$\boldsymbol{a}_i \leftarrow \mathbb{Z}_q^n$, $e_i \leftarrow \chi$, and setting $b_i = \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i \bmod q$. The $\mathcal{DLWE}_{n,q,\chi}$ assumption is that $\mathcal{DLWE}_{n,q,\chi}$ is infeasible.

Recall that GapSVP_γ is the promise problem to distinguish between the case in which the lattice has a vector shorter than $r \in \mathbb{Q}$, and the case in which all the lattice vectors are greater than $\gamma \cdot r$. SIVP_γ is the problem to find the set of short linearly independent vectors in a lattice. $\mathcal{DLWE}_{n,q,\chi}$ has reductions to the standard lattice assumptions as follows. These reductions take χ to be a discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\alpha q}$ (that is centered around 0 and has parameter αq for some $\alpha < 1$).

Corollary 2.1 [8] *Let $q = q(n) \in \mathbb{N}$ be a power of primes $q = p^r$ or a product of distinct prime numbers $q = \prod_i q_i$ ($q_i = \text{poly}(n)$ for all i), and let $\alpha \geq \frac{\sqrt{n}}{q}$. If there exists an efficient algorithm that solves (average-case) $\mathcal{DLWE}_{n,q,\mathcal{D}_{\mathbb{Z},\alpha q}}$,*

- there exists an efficient quantum algorithm that can solve $\text{GapSVP}_{\tilde{O}\left(\frac{n}{\alpha}\right)}$ and $\text{SIVP}_{\tilde{O}\left(\frac{n}{\alpha}\right)}$ in the worst-case for any n-dimensional lattices.
- if in addition we have $q \geq \tilde{O}\left(2^{\frac{n}{2}}\right)$, there exists an efficient classical algorithm that can solve $\text{GapSVP}_{\tilde{O}\left(\frac{n}{\alpha}\right)}$ in the worst-case for any n-dimensional lattices.

On this basis, one can select n secret vectors $\boldsymbol{s}$ to create the matrix $\mathbf{S} \in \mathbb{Z}_q^{m \times n}$, which leads to the definition of the n-secret LWE assumption.

Definition 2.4 (n-secret LWE) [17] For a security parameter λ, let $n = n(\lambda)$ and $m = m(\lambda)$ be integer dimension, let $q = q(\lambda) \geq 2$ be an integer modulus, and let $\chi = \chi(\lambda)$ be an error distribution over $\mathbb{Z}$. $n\text{-}\mathcal{LWE}_{n,m,q,\chi}$ is the problem to distinguish the following two distributions: In the first distribution, a tuple $(\mathbf{A}, \mathbf{B})$ is sampled from uniform over $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^{n \times n}$; In the second distribution, $\mathbf{S} \leftarrow \mathbb{Z}_q^{m \times n}$ and then a tuple $(\mathbf{A}, \mathbf{B})$ is sampled by sampling $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{E} \leftarrow \chi^{n \times n}$, and setting $\mathbf{B} = (\mathbf{AS} + \mathbf{E}) \bmod q$. The $n\text{-}\mathcal{LWE}_{n,m,q,\chi}$ assumption is that $n\text{-}\mathcal{LWE}_{n,m,q,\chi}$ is infeasible.

The security of our scheme is based on the n-secret LWE assumption, and we describe our construction of fully homomorphic encryption in detail below.

3 Fully homomorphic encryption scheme for matrix

This section mainly introduces the fully homomorphic encryption scheme for matrices, provides a detailed proof of its decryption correctness, homomorphic properties, and noise analysis. Furthermore, we utilize modulus switching and key switching technologies in our scheme to refresh the ciphertext and achieve noise reduction. Finally, we also provide a security analysis of our scheme.

3.1 Our scheme—GMS

Let λ be the security parameter. The plaintext space $\mathcal{M}$ is the set of $n \times n$-dimensional matrix, where each component of the matrix is an element in the set $\{-t, -(t-1), \ldots, -1, 0, 1, \ldots, t-1, t\}$. This means that the norm of $\mathcal{M}$ is less than or equal to t, denoted as $\|\mathcal{M}\| \leq t$. where t is a positive integer. Let q be a large integer, $q \gg t$, $a = \lceil \log_b q \rceil$, $N = na$, and $M = ma$. We also have integer values for α, β, and η that satisfy $\alpha\beta = t$, $\alpha < \beta$ and β and η are invertible modulo q, satisfying $\beta^{-1} \bmod q = \eta$, which means that $\beta\eta \bmod q = 1$.

- **KeyGen**$(1^\lambda, pk, sk)$: Randomly and uniformly choose an $n \times m$-dimensional matrix $\mathbf{A} \leftarrow \mathbb{Z}_t^{n\times m}$. Randomly and secretly choose two $m \times n$-dimensional matrix $\mathbf{S}_1, \mathbf{S}_2 \leftarrow \mathbb{Z}_q^{m\times n}$, where each component is in the set $\{-1, 0, 1\}$. The error matrix $\mathbf{E} \leftarrow \mathbb{Z}_q^{n\times n}$, and each component of $\mathbf{E}$ follows the error distribution χ. Compute $\mathbf{B} = \left(\mathbf{AS}_1 + \mathbf{E}\right) \bmod q$, and let $\mathbf{D} = (\alpha\mathbf{B}|\mathbf{A}) \in \mathbb{Z}_q^{n\times(n+m)}$, $\mathbf{T}_1 = \alpha\left(\mathbf{S}_1 + \mathbf{S}_2\right) \in \mathbb{Z}_q^{m\times n}$. Set $\mathbf{V} = \left(\mathbf{I}_n|\mathbf{O}_{n\times m}\right) \in \mathbb{Z}_q^{n\times(n+m)}$, $\mathbf{S} = \begin{pmatrix} \beta\mathbf{I}_n \\ -t\mathbf{S}_1 \end{pmatrix} \in \mathbb{Z}_q^{(n+m)\times n}$, $\mathbf{T} = \begin{pmatrix} \mathbf{I}_n \\ -\mathbf{T}_1 \end{pmatrix} \in \mathbb{Z}_q^{(n+m)\times n}$, Then we have $\mathbf{DS} = t\mathbf{E}$, $\mathbf{VS} = \beta\mathbf{I}_n$, $\mathbf{DT} = \alpha\left(\mathbf{E} - \mathbf{AS}_2\right)$, $\mathbf{VT} = \mathbf{I}_n$. The secret key $sk = \mathbf{S}$, the public key $pk = (\mathbf{T}, \mathbf{V}, \mathbf{D})$.
- **Encrypt**$(pk, \mathbf{M})$: Given the plaintext $\mathbf{M} \in \mathcal{M}$, randomly and uniformly choose an $(N+M) \times n$-dimensional matrix $\mathbf{R}$, each component of $\mathbf{R}$ is an element in the set $\{-1, 0, 1\}$, i.e., $\mathbf{R} \leftarrow \{-1, 0, 1\}^{(N+M)\times n}$. Output the ciphertext $\mathbf{C} = (\mathbf{GTMV} + \mathbf{RD}) \bmod q$, where $\mathbf{G} = \mathbf{I}_{n+m} \otimes \boldsymbol{g}$ is a gadget matrix.
- **Decrypt**$(sk, \mathbf{C})$: Output $\mathbf{M}' = \left(G^{-1}(\eta\mathbf{V})\mathbf{CS} \bmod q\right) \bmod t$.
- **Eval-Add**$(pk, \mathbf{C}_1, \mathbf{C}_2)$: Given ciphertexts $\mathbf{C}_1$ and $\mathbf{C}_2$, output $\mathbf{C}_{add} = \left(\mathbf{C}_1 + \mathbf{C}_2\right) \bmod q$.
- **Eval-Mult**$(pk, \mathbf{C}_1, \mathbf{C}_2)$: Given ciphertexts $\mathbf{C}_1$ and $\mathbf{C}_2$, output $\mathbf{C}_{mult} = G^{-1}\left(\mathbf{C}_1\right)\mathbf{C}_2 \bmod q$.
- **Refresh**$(\mathbf{C}, ksk, q_1, q_2)$: Given ciphertext $\mathbf{C}$ and modulus $q_1 > q_2$, run $\mathbf{C}' \leftarrow$ **ModSwitch**$(\mathbf{C})$, where $\mathbf{C}'$ is encrypted under the same key with $\mathbf{C}$ for a different modulus q_2. Then run $\mathbf{C}^\varepsilon \leftarrow$ **KeySwt**$(\mathbf{C}', ksk)$, where $\mathbf{C}^\varepsilon$ is encrypted under a different key (pk', sk') for the same modulus with $\mathbf{C}'$.

3.2 Correctness of decryption

In this section, we mainly verify the correctness of decryption. For $\mathbf{C}$, which is the encryption of $\mathbf{M}$, consider the decryption process of $\mathbf{C}$:

$$\begin{aligned}\mathbf{Decrypt}(sk,\mathbf{C}) &= \left(G^{-1}(\eta\mathbf{V})\mathbf{CS} \bmod q\right) \bmod t\\ &= \left(G^{-1}(\eta\mathbf{V})(\mathbf{GTMV}+\mathbf{RD})\mathbf{S} \bmod q\right) \bmod t\\ &= \left(\left(\eta\mathbf{VTMVS}+G^{-1}(\eta\mathbf{V})\mathbf{RDS}\right) \bmod q\right) \bmod t\\ &= \left(\eta\beta\mathbf{M}+tG^{-1}(\eta\mathbf{V})\mathbf{RE} \bmod q\right) \bmod t\\ &= \left(\mathbf{M}+tG^{-1}(\eta\mathbf{V})\mathbf{RE}\right) \bmod t\\ &= \mathbf{M}.\end{aligned} \tag{1}$$

It is easy to see that decrypting the ciphertext $\mathbf{C}$ yields the plaintext $\mathbf{M}$. In other words, the decryption is correct.

3.3 Homomorphic properties

In this section, we first define the operations of homomorphic addition and homomorphic multiplication in our scheme. We then proceed to verify their homomorphic properties.

- Homomorphic addition: $\mathbf{C}_{add} = \left(\mathbf{C}_1 + \mathbf{C}_2\right) \bmod q$.

$$\begin{aligned}\mathbf{C}_{add} &= \left(\mathbf{C}_1+\mathbf{C}_2\right) \bmod q\\ &= \left(\mathbf{GTM}_1\mathbf{V}+\mathbf{R}_1\mathbf{D}\right)+\left(\mathbf{GTM}_2\mathbf{V}+\mathbf{R}_2\mathbf{D}\right) \bmod q\\ &= \left(\mathbf{GT}\left(\mathbf{M}_1+\mathbf{M}_2\right)\mathbf{V}+\left(\mathbf{R}_1+\mathbf{R}_2\right)\mathbf{D}\right) \bmod q.\end{aligned} \tag{2}$$

Decrypt $\mathbf{C}_{add}$, the specific decryption process is as follows:

$$\begin{aligned}\mathbf{Decrypt}\left(sk,\mathbf{C}_{add}\right) &= \left(G^{-1}(\eta\mathbf{V})\mathbf{C}_{add}\mathbf{S} \bmod q\right) \bmod t\\ &= \left(\left(\eta\mathbf{VT}\left(\mathbf{M}_1+\mathbf{M}_2\right)\mathbf{VS}+G^{-1}(\eta\mathbf{V})\left(\mathbf{R}_1+\mathbf{R}_2\right)\mathbf{DS}\right) \bmod q\right) \bmod t\\ &= \left(\left(\eta\beta\left(\mathbf{M}_1+\mathbf{M}_2\right)+tG^{-1}(\eta\mathbf{V})\left(\mathbf{R}_1+\mathbf{R}_2\right)\mathbf{E}\right) \bmod q\right) \bmod t\\ &= \left(\left(\mathbf{M}_1+\mathbf{M}_2\right)+tG^{-1}(\eta\mathbf{V})\left(\mathbf{R}_1+\mathbf{R}_2\right)\mathbf{E}\right) \bmod t\\ &= \mathbf{M}_1+\mathbf{M}_2.\end{aligned} \tag{3}$$

Easy to know, $\mathbf{C}_{add}$ corresponds to the encryption of the plaintext $\mathbf{M}_1+\mathbf{M}_2$.

- Homomorphic multiplication $\mathbf{C}_{mult} = G^{-1}\left(\mathbf{C}_1\right)\mathbf{C}_2 \bmod q$.

$$\begin{aligned}\mathbf{C}_{mult} &= G^{-1}\left(\mathbf{C}_1\right)\mathbf{C}_2 \bmod q\\ &= G^{-1}\left(\mathbf{C}_1\right)\left(\mathbf{GTM}_2\mathbf{V}+\mathbf{R}_2\mathbf{D}\right) \bmod q\\ &= \left(\left(\mathbf{GTM}_1\mathbf{V}+\mathbf{R}_1\mathbf{D}\right)\mathbf{TM}_2\mathbf{V}+G^{-1}\left(\mathbf{C}_1\right)\mathbf{R}_2\mathbf{D}\right) \bmod q\\ &= \left(\mathbf{GT}\left(\mathbf{M}_1\mathbf{M}_2\right)\mathbf{V}+\alpha\mathbf{R}_1\left(\mathbf{E}-\mathbf{AS}_2\right)\mathbf{M}_2\mathbf{V}+G^{-1}\left(\mathbf{C}_1\right)\mathbf{R}_2\mathbf{D}\right) \bmod q.\end{aligned} \tag{4}$$

Decrypt $\mathbf{C}_{mult}$, the specific decryption process is as follows:

$$\begin{aligned}&\mathbf{Decrypt}\left(sk,\mathbf{C}_{mult}\right)\\ &= \left(G^{-1}(\eta\mathbf{V})\mathbf{C}_{mult}\mathbf{S} \bmod q\right) \bmod t\\ &= \left(\left(\eta\beta\mathbf{M}_1\mathbf{M}_2+\alpha\beta G^{-1}(\eta\mathbf{V})\mathbf{R}_1\left(\mathbf{E}-\mathbf{AS}_2\right)\mathbf{M}_2+tG^{-1}(\eta\mathbf{V})G^{-1}\left(\mathbf{C}_1\right)\mathbf{R}_2\mathbf{E}\right) \bmod q\right) \bmod t\\ &= \left(\mathbf{M}_1\mathbf{M}_2+tG^{-1}(\eta\mathbf{V})\mathbf{R}_1\left(\mathbf{E}-\mathbf{AS}_2\right)\mathbf{M}_2+tG^{-1}(\eta\mathbf{V})G^{-1}\left(\mathbf{C}_1\right)\mathbf{R}_2\mathbf{E}\right) \bmod t\\ &= \mathbf{M}_1\mathbf{M}_2.\end{aligned} \tag{5}$$

Easy to know, $\mathbf{C}_{mult}$ corresponds to the encryption of the plaintext $\mathbf{M}_1\mathbf{M}_2$.

On this basis, we can further optimize the process of homomorphic multiplication. $G^{-1}(\mathbf{C}_1)$ is a $(N+M)\times(N+M)$-dimensional matrix, $\mathbf{C}_2$ is a $(N+M)\times(n+m)$ -dimensional matrix, which can be expressed as

$$G^{-1}(\mathbf{C}_1)=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1,N+M} \\ a_{21} & a_{22} & \cdots & a_{2,N+M} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N+M,1} & a_{N+M,2} & \cdots & a_{N+M,N+M} \end{pmatrix}, \mathbf{C}_2=\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1,n+m} \\ b_{21} & b_{22} & \cdots & b_{2,n+m} \\ \vdots & \vdots & \vdots & \vdots \\ b_{N+M,1} & b_{N+M,2} & \cdots & b_{N+M,n+m} \end{pmatrix}. \tag{6}$$

Rotate these two matrices by their diagonal elements to obtain the matrices $\mathbf{A}$ and $\mathbf{B}$. The specific rotation operation is as follows.

$$\mathbf{A}=\text{Rotate}\left(G^{-1}(\mathbf{C}_1)\right)=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1,N+M-1} & a_{1,N+M} \\ a_{22} & a_{23} & \cdots & a_{2,N+M} & a_{21} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{N+M,N+M} & a_{N+M,1} & \cdots & a_{N+M,N+M-2} & a_{N+M,N+M-1} \end{pmatrix}, \tag{7}$$

$$\mathbf{B}=\text{Rotate}\left(\mathbf{C}_2\right)=\begin{pmatrix} b_{11} & b_{22} & \cdots & b_{n+m,n+m} \\ b_{21} & b_{32} & \cdots & b_{n+m+1,n+m} \\ \vdots & \vdots & \vdots & \vdots \\ b_{N+M-1,1} & b_{N+M,2} & \cdots & b_{n+m-2,n+m} \\ b_{N+M,1} & b_{12} & \cdots & b_{n+m-1,n+m} \end{pmatrix}. \tag{8}$$

Then use the $(N+M)$-dimensional column vector $\boldsymbol{a}_i\ (1\le i\le N+M)$ to represent each column element of the matrix $\mathbf{A}$, and use the $(N+M)$-dimensional row vector $\boldsymbol{b}_j^T\ (1\le j\le N+M)$ to represent each row element of the matrix $\mathbf{B}$, that is, $\mathbf{A}=\left(\boldsymbol{a}_1,\boldsymbol{a}_2,\ldots,\boldsymbol{a}_{N+M}\right), \mathbf{B}=\left(\boldsymbol{b}_1^T,\boldsymbol{b}_2^T,\ldots,\boldsymbol{b}_{N+M}^T\right)^T$. Let the $(N+M)\times(n+m)$-dimensional matrix $\mathbf{A}_i\ (1\le i\le N+M)$ and $\mathbf{B}_j\ (1\le j\le N+M)$ be pressed as follows:

$$\begin{aligned}\mathbf{A}_1=&\left(\boldsymbol{a}_1,\boldsymbol{a}_2,\ldots,\boldsymbol{a}_{n+m}\right), \mathbf{A}_2=\left(\boldsymbol{a}_2,\boldsymbol{a}_3,\ldots,\boldsymbol{a}_{n+m+1}\right), \mathbf{A}_3=\left(\boldsymbol{a}_3,\boldsymbol{a}_4,\ldots,\boldsymbol{a}_{n+m+2}\right),\\ &\ldots, \mathbf{A}_{N+M-1}=\left(\boldsymbol{a}_{N+M-1},\boldsymbol{a}_{N+M},\ldots,\boldsymbol{a}_{n+m-2}\right), \mathbf{A}_{N+M}=\left(\boldsymbol{a}_{N+M},\boldsymbol{a}_1,\ldots,\boldsymbol{a}_{n+m-1}\right).\end{aligned} \tag{9}$$

$$\begin{aligned}\mathbf{B}_1=&\left(\boldsymbol{b}_1^T,\boldsymbol{b}_2^T,\ldots,\boldsymbol{b}_{N+M-1}^T,\boldsymbol{b}_{N+M}^T\right)^T, \mathbf{B}_2=\left(\boldsymbol{b}_2^T,\boldsymbol{b}_3^T,\ldots,\boldsymbol{b}_{N+M}^T,\boldsymbol{b}_1^T\right)^T, \mathbf{B}_3=\left(\boldsymbol{b}_3^T,\boldsymbol{b}_4^T,\ldots,\boldsymbol{b}_1^T,\boldsymbol{b}_2^T\right)^T,\\ &\ldots, \mathbf{B}_{N+M}=\left(\boldsymbol{b}_{N+M}^T,\boldsymbol{b}_1^T,\ldots,\boldsymbol{b}_{N+M-2}^T,\boldsymbol{b}_{N+M-1}^T\right)^T.\end{aligned} \tag{10}$$

So the homomorphic multiplication process can be transformed into

$$G^{-1}(\mathbf{C}_1)\cdot\mathbf{C}_2=\sum_{i=1}^{N+M}\mathbf{A}_i\odot\mathbf{B}_i=\sum_{i=1}^{N+M}\text{Rotate}(\mathbf{A},0,-i)\odot\text{Rotate}(\mathbf{B},1,-i), \tag{11}$$

where $\odot$ represents the element-wise multiplication of the matrix, $\text{Rotate}(\mathbf{A},0,-i)$ indicates rotating each element of the matrix $\mathbf{A}$ by i positions to the left, and $\text{Rotate}(\mathbf{B},1,-i)$ indicates rotating each element of the matrix $\mathbf{B}$ by i positions

upwards. This can convert one ciphertext matrix multiplication into $O(n)$ rotations and $O(n)$ constant multiplications, which is more efficient.

3.4 Noise analysis

The noise in the scheme arises from the modular operation during the decryption process. let

$$\mathcal{N}(\mathbf{C}) = G^{-1}(\eta\mathbf{V})\mathbf{CS} \bmod q - \mathbf{M} = tG^{-1}(\eta\mathbf{V})\mathbf{RE}. \tag{12}$$

- Initial noise: $\|\mathcal{N}(\mathbf{C})\| = \left\|tG^{-1}(\eta\mathbf{V})\mathbf{RE}\right\| \le t(N+M)n\left\|G^{-1}(\eta\mathbf{V})\right\|\|\mathbf{R}\|\|\mathbf{E}\| \le t(N+M)nb\|\mathbf{E}\|$.
- Additional noise: $\left\|\mathcal{N}(\mathbf{C}_{add})\right\| = \left\|\mathcal{N}(\mathbf{C}_1)\right\| + \left\|\mathcal{N}(\mathbf{C}_2)\right\| \le 2t(N+M)nb\|\mathbf{E}\|$.
- Multiplication noise:

$$\begin{aligned}\left\|\mathcal{N}(\mathbf{C}_{mul})\right\| &= \left\|tG^{-1}(\eta\mathbf{V})\left(\mathbf{R}_1\left(\mathbf{E}-\mathbf{AS}_2\right)\mathbf{M}_2 + G^{-1}\left(\mathbf{C}_1\right)\mathbf{R}_2\mathbf{E}\right)\right\| \\ &\le t(N+M)\left\|G^{-1}(\eta\mathbf{V})\right\|\left(\left\|\mathbf{R}_1\left(\mathbf{E}-\mathbf{AS}_2\right)\mathbf{M}_2\right\| + \left\|G^{-1}\left(\mathbf{C}_1\right)\mathbf{R}_2\mathbf{E}\right\|\right) \\ &\le t(N+M)b\left(n^2\|\mathbf{R}_1\|\|\mathbf{E}-\mathbf{AS}_2\|\|\mathbf{M}_2\| + (N+M)n\left\|G^{-1}\left(\mathbf{C}_1\right)\right\|\|\mathbf{R}_2\|\|\mathbf{E}\|\right) \\ &\le t(N+M)b\left(n^2t\left(\|\mathbf{E}\| + m\|\mathbf{A}\|\left\|\mathbf{S}_2\right\|\right) + (N+M)nb\|\mathbf{E}\|\right) \\ &\le t(N+M)b\left(\left(n^2+m\right)t + (N+M)nb\right)\|\mathbf{E}\|.\end{aligned} \tag{13}$$

In summary, both initial noise and additional noise can be expressed as $O\left(n^2tbae\right)$, while multiplication noise can be expressed as $O\left(n^3t^2b^2a^2e\right)$, where e represents the maximum value of the norm of error matrix $\mathbf{E}$, i.e., $\|\mathbf{E}\| \le e$. Therefore, the upper bound N of noise can be expressed as $N = O\left(n^3t^2b^2a^2e\right)$, and the parameter selection of the scheme is closely related to N and needs to satisfy $|N| < \frac{q}{2}$.

3.5 Ciphertext refreshing

According to the previous sections, it is evident that the accuracy of our scheme primarily relies on the modular operation mod$t$. Specifically, the noise component should be $\le \frac{q}{2}$. Based on the aforementioned analysis of noise, the maximum limit of noise can be achieved by selecting appropriate parameters. In order to enhance the efficiency of secure matrix multiplication and reduce the inclusion of redundant items in the operation, we introduce a ciphertext refreshing program in this section.

Similar to the ciphertext refreshing technique used in BGV [11], modulus switching technology is employed to convert the ciphertext from a large modulus to a small modulus. Subsequently, key switching technology is applied to change the ciphertext under different keys. This entire refreshing process transforms the ciphertext into another ciphertext under a different key and a smaller modulus, while still representing the same plaintext. The newly introduced modulus switching and key switching technologies are described below.

3.5.1 Modulus switching

Similar to the existing modulus switching technology, we need to define a mapping from a large modulus to a small modulus in order to implement the modulus switching process **ModSwitch**. The specific steps are as follows:

- Define a mapping from a large modulus q_1 to a small modulus q_2: For any integer a, $[a]_{q_1 \to q_2} = \left\lfloor \frac{q_2}{q_1} a \right\rfloor + r$, where $r \in \{0, 1\}$.
- For the ciphertext $\mathbf{C} = \left(c_{ij}\right), 1 \le i \le N + M, 1 \le j \le n + m$, the modulus mapping of each component can be defined as a pair. The specific definition of modulus switching is $\mathbf{ModSwitch}(\mathbf{C}) = \left([c_{ij}]_{q_1 \to q_2}\right)$.

Let $\mathbf{C}' = \mathbf{ModSwitch}(\mathbf{C}) = \left([c_{ij}]_{q_1 \to q_2}\right)$, then $\mathbf{C}' \approx \frac{q_2}{q_1}\mathbf{C}$, the noise also becomes the original $\frac{q_2}{q_1}$. Modulus switching, denoted as **ModSwitch**, has the effect of reducing noise. The ciphertext $\mathbf{C}$, which is encrypted using the modulo q_1, is transformed into a new ciphertext $\mathbf{C}'$ using a different modulo q_2. In this case, it represents the same plaintext $\mathbf{M}$, but the noise is reduced by approximately $\frac{q_2}{q_1}$.

3.5.2 Key switching

Similar to the existing key switching, our program also consists of two steps: first, generating a key switching key *ksk*, and then using this *ksk* to publicly implement the ciphertext conversion. Our goal is to convert a ciphertext matrix $\mathbf{C}$ determined by the key $\left(pk_1, sk_1\right)$ to a ciphertext matrix $\mathbf{C}'$ determined by the key $\left(pk_2, sk_2\right)$.

- $\mathbf{KeySwtGen}(pk_1, pk_2, sk_1, sk_2)$: Define one key pair is $pk_1 = \left(\mathbf{T}_1, \mathbf{V}, \mathbf{D}_1\right)$, $sk_1 = \mathbf{S}_{(1)}$ and another key pair is $pk_2 = \left(\mathbf{T}_2, \mathbf{V}, \mathbf{D}_2\right)$, $sk_2 = \mathbf{S}_{(2)}$. The process of generating a key switching key is as follows:

  1. Define $\boldsymbol{g} = \left(1, b, b^2, \ldots, b^{a-1}\right)^T$, and $\mathbf{G} = \mathbf{I}_{m+n} \otimes \boldsymbol{g} \in \mathbb{Z}^{(M+N)\times(m+n)}$.
  2. Randomly and uniformly choose a matrix $\mathbf{W} \leftarrow \{-1, 0, 1\}^{(N+M)\times n}$.
  3. Output key switching key $ksk = \left(\mathbf{G} + \mathbf{W}\mathbf{D}_1\right)\mathbf{S}_{(1)}\mathbf{V} \in \mathbb{Z}_q^{(N+M)\times n}$.

- $\mathbf{KeySwt}(\mathbf{C}_1, ksk)$: Given a ciphertext $\mathbf{C} \in \mathbb{Z}_q$ and a key switching key $ksk \in \mathbb{Z}_q^{(N+M)\times(n+m)}$, define $\mathbf{Y} = \mathbf{G}\mathbf{T}_2 G^{-1}(\eta\mathbf{V}) G^{-1}(\mathbf{C}) \in \mathbb{Z}_q^{(N+M)\times(N+M)}$, and output $\mathbf{C}' = \mathbf{Y} \cdot ksk \bmod q \in \mathbb{Z}_q$.

3.5.3 Correctness of modulus switching and key switching

The correctness of modulus switching is relatively easy and will not be verified in detail here. The main consideration is the correctness of key switching technology. First, calculate

$$\begin{aligned}\mathbf{C}' &= \mathbf{Y} \cdot ksk \bmod q \\ &= \mathbf{GT}_2 G^{-1}(\eta\mathbf{V}) G^{-1}(\mathbf{C})\big(\mathbf{G} + \mathbf{WD}_1\big)\mathbf{S}_{(1)}\mathbf{V} \bmod q \\ &= \mathbf{GT}_2 G^{-1}(\eta\mathbf{V})\mathbf{CS}_{(1)}\mathbf{V} + \mathbf{GT}_2 G^{-1}(\eta\mathbf{V}) G^{-1}(\mathbf{C})\mathbf{WD}_1\mathbf{S}_{(1)}\mathbf{V} \bmod q \\ &= \mathbf{GT}_2\big(\eta\beta\mathbf{M} + tG^{-1}(\eta\mathbf{V})\mathbf{RE}_1\big)\mathbf{V} + t\mathbf{GT}_2 G^{-1}(\eta\mathbf{V}) G^{-1}(\mathbf{C})\mathbf{WE}_1\mathbf{V} \bmod q \\ &= \mathbf{GT}_2\mathbf{MV} + t\big(G^{-1}(\eta\mathbf{V})\mathbf{RE}_1\mathbf{V} + \mathbf{GT}_2 G^{-1}(\eta\mathbf{V}) G^{-1}(\mathbf{C})\mathbf{WE}_1\mathbf{V}\big).\end{aligned} \tag{14}$$

We know that the ciphertext $\mathbf{C}'$ is the encryption of plaintext $\mathbf{M}$. Its noise is $t\big(G^{-1}(\eta\mathbf{V})\mathbf{RE}_1\mathbf{V} + \mathbf{GT}_2 G^{-1}(\eta\mathbf{V}) G^{-1}(\mathbf{C})\mathbf{WE}_1\mathbf{V}\big)$, which is still a multiple of t, and it has the same form as the noise. This noise, which can be eliminated by subsequent operations using modulo t, can be decrypted correctly. Therefore, the ciphertext $\mathbf{C}'$ can be decrypted correctly, and the homomorphic evaluation can be performed accurately.

3.6 Security analysis

Here, we prove the following theorem, which asserts the security of our scheme.

Theorem 3.1 *Assuming DLWE (Definition 2.3) and n-secret LWE (Definition 2.4), the security of our scheme only depends on n-secret LWE assumption, and the fully homomorphic encryption scheme we constructed satisfies IND-CPA security* [29].

Proof To prove the security, we need to prove the indistinguishability of the following two distributions.

Distribution $\mathcal{D}_1$: $\mathcal{D}_1 = \{\mathbf{C} \leftarrow (\mathbf{GTMV} + \mathbf{RD}) \bmod q | pk = (\mathbf{T}, \mathbf{V}, \mathbf{D}), sk = \mathbf{S}\}$.

Distribution $\mathcal{D}_2$: $\mathcal{D}_2 = \left\{\mathbf{C} \leftarrow \mathbb{Z}_q^{(N+M)\times(n+m)} | pk = (\mathbf{T}, \mathbf{V}, \mathbf{D}), sk = \mathbf{S}\right\}$.

We can see that $\mathcal{D}_1$ and $\mathcal{D}_2$ represent the adversary's views when the plaintext $\mathbf{M}$ is encrypted and randomly chosen, respectively. From the **keyGen** process, we have $\mathbf{D} = (\alpha\mathbf{B}|\mathbf{A})$. Therefore, to demonstrate that the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable, we need to establish that the following two distributions are computationally indistinguishable:

Distribution $\mathcal{D}_1'$: $\mathcal{D}_1' = \{\mathbf{C} \leftarrow (\mathbf{GTMV} + \mathbf{R}(\alpha\mathbf{B}|\mathbf{A})) \bmod q | \mathbf{B} = (\mathbf{AS} + \mathbf{E}) \bmod q\}$.

$\mathcal{D}_1' = \{\mathbf{C} \leftarrow (\mathbf{GTMV} + \mathbf{R}(\alpha\mathbf{B}|\mathbf{A})) \bmod q | \mathbf{B} = (\mathbf{AS} + \mathbf{E}) \bmod q\}$.

Distribution $\mathcal{D}_2'$: $\mathcal{D}_2' = \left\{\mathbf{C} \leftarrow \mathbb{Z}_q^{(N+M)\times(n+m)}\right\}$.

From the above discussion, it suffices to prove Lemma 3.1 in the following to complete the proof of Theorem 3.1.

Lemma 3.1 *Distributions $\mathcal{D}_1'$ and $\mathcal{D}_2'$ are computationally indistinguishable under the hardness assumption of DLWE and n-secret LWE.*

Proof We prove the lemma via the following hybrids.

$\mathcal{G}_0$: This is same as $\mathcal{D}_1'$.

$\mathcal{G}_1$: In this hybrid, the challenger computes $\mathbf{C}$ as $\mathbf{C} = (\mathbf{GTMV} + \mathbf{R}(\alpha\mathbf{B}|\mathbf{A})) \bmod q$, where $\mathbf{B} = \big(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\big)$, and $\boldsymbol{b}_i = \big(\mathbf{A}\boldsymbol{s}_i + \boldsymbol{e}_i\big) \bmod q$, where each component of $\mathbf{A}$ is

selected from a random uniform distribution, and each component of $\boldsymbol{e}_i$ is selected from an error distribution χ.

$\mathcal{G}_2$: The challenger computes $\mathbf{C} = (\mathbf{GTMV} + \mathbf{U}) \bmod q$, where $\mathbf{U} \leftarrow \mathbb{Z}_q^{(N+M)\times(n+m)}$.

$\mathcal{G}_3$: In this hybrid, the challenger randomly and uniformly selects $\mathbf{C} \leftarrow \mathbb{Z}_q^{(N+M)\times(n+m)}$.

It is easy to see that the distribution of G_3 is the same as the distribution $\mathcal{D}_2^{'}$. We prove the indistinguishability of the distribution in the following way.

First, prove that $\mathcal{G}_0 \equiv \mathcal{G}_1$. Since only the selection method of $\mathbf{B}$ is different in $\mathcal{G}_0$ and $\mathcal{G}_1$, it is easy to see that $\mathbf{B} = (\mathbf{AS} + \mathbf{E}) \bmod q$ represents the standard DLWE assumption. $\boldsymbol{b}_i = \left(\mathbf{A}\boldsymbol{s}_i + \boldsymbol{e}_i\right) \bmod q$ represents the spliced form of the n secret vectors $\boldsymbol{s}_i$, ensuring the equivalence of $\mathcal{G}_0$ and $\mathcal{G}_1$.

Then prove that $\mathcal{G}_1 \approx_c \mathcal{G}_2$ due to DLWE. Since only the second half of the random selection in $\mathcal{G}_1$ and $\mathcal{G}_2$ is different, we know that the DLWE distribution and the uniform random distribution are computationally indistinguishable. In other words, $\{(\mathbf{B}, \mathbf{A})\}$ and $\mathbb{Z}_q^{(N+M)\times(n+m)}$ are computationally indistinguishable. By the definition of the cipher, $\mathbf{R} \leftarrow \{-1, 0, 1\}^{(N+M)\times(n+m)}$, so the distributions of $\mathbf{R}(\alpha\mathbf{B}, \mathbf{A})$ and $\mathbf{U}$ are computationally indistinguishable. Therefore, the distributions of $\mathcal{G}_1$ and $\mathcal{G}_2$ are computationally indistinguishable under the standard DLWE assumption.

Next, it is proven that $\mathcal{G}_2 \approx_s \mathcal{G}_3$. Since the public keys $\mathbf{T}$ and $\mathbf{V}$ are both public, and $\mathbf{G}$ is the gadget matrix, $\mathbf{GTMV}$ can be treated as a constant. Therefore, $(\mathbf{GTMV} + \mathbf{U}) \bmod q$ and $\mathbf{C} \leftarrow \mathbb{Z}_q^{(N+M)\times(n+m)}$ are statistically indistinguishable, where $\mathbf{U} \leftarrow \mathbb{Z}_q^{(N+M)\times(n+m)}$. means that the distributions of $\mathcal{G}_2$ and $\mathcal{G}_3$ are statistically indistinguishable.

Combined with the above analysis, it is easy to see that the distributions of $\mathcal{G}_0$ and $\mathcal{G}_3$ are computationally indistinguishable.

In summary, the distributions $\mathcal{D}_1^{'}$ and $\mathcal{D}_2^{'}$ are computationally indistinguishable, implying that the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are also computationally indistinguishable. This implies that our scheme is IND-CPA if the n-secret LWE problem is computationally hard.

Thus, if the IND-CPA security of the underlying fully homomorphic encryption scheme can be guaranteed, it implies that the ciphertexts of any two messages are computationally indistinguishable. Since all computations in the public cloud are performed on encrypted data, the cloud cannot access any information from the encrypted data. Therefore, we can ensure the confidentiality of the data when using semi-honest servers for secure outsourced computation.

4 Secure matrix multiplication based on GMS

Since our scheme GMS is an asymmetric encryption scheme for matrices, the secure matrix multiplication procedure can be easily implemented. GMS can achieve rectangular matrix multiplication $\mathbf{A}_{m\times n} \cdot \mathbf{B}_{n\times l}$ by padding with zeros.

This can be elaborated upon by considering the following three cases: (1) $\max\{m,n,l\}=m$; (2) $\max\{m,n,l\}=n$; (3) $\max\{m,n,l\}=l$. The padding operation is as follows.

(1) If $\max\{m,n,l\}=m$, then first fill the matrices $\mathbf{A}$ and $\mathbf{B}$ into a square matrix, that is, $\mathbf{A}'_{m\times m}=\begin{pmatrix}\mathbf{A}_{m\times n} & \mathbf{O}_{m\times(m-n)}\end{pmatrix}$, $\mathbf{B}'_{m\times m}=\begin{pmatrix}\mathbf{B}_{n\times l} & \mathbf{O}_{n\times(m-l)}\\ \mathbf{O}_{(m-n)\times l} & \mathbf{O}_{(m-n)\times(m-l)}\end{pmatrix}$. We have

$$\mathbf{A}'\cdot\mathbf{B}'=\begin{pmatrix}\mathbf{A} & \mathbf{O}\end{pmatrix}\cdot\begin{pmatrix}\mathbf{B} & \mathbf{O}\\ \mathbf{O} & \mathbf{O}\end{pmatrix}=\begin{pmatrix}\mathbf{A}\cdot\mathbf{B}_{m\times l} & \mathbf{O}_{m\times(m-l)}\end{pmatrix}.$$

So $\mathrm{Enc}\begin{pmatrix}\mathbf{A}\cdot\mathbf{B} & \mathbf{O}\end{pmatrix}$ can be obtained by passing $\mathrm{Enc}\begin{pmatrix}\mathbf{A} & \mathbf{O}\end{pmatrix}$ and $\mathrm{Enc}\begin{pmatrix}\mathbf{B} & \mathbf{O}\\ \mathbf{O} & \mathbf{O}\end{pmatrix}$ through the homomorphic multiplication operation, where Enc represents the encryption algorithm of our scheme.

(2) If $\max\{m,n,l\}=n$, then first fill the matrices $\mathbf{A}$ and $\mathbf{B}$ into a square matrix, that is, $\mathbf{A}'_{n\times n}=\begin{pmatrix}\mathbf{A}_{m\times n}\\ \mathbf{O}_{(n-m)\times n}\end{pmatrix}$, $\mathbf{B}'_{n\times n}=\begin{pmatrix}\mathbf{B}_{n\times l} & \mathbf{O}_{n\times(n-l)}\end{pmatrix}$. We have

$$\mathbf{A}'\cdot\mathbf{B}'=\begin{pmatrix}\mathbf{A}\\ \mathbf{O}\end{pmatrix}\cdot\begin{pmatrix}\mathbf{B} & \mathbf{O}\end{pmatrix}=\begin{pmatrix}\mathbf{A}\cdot\mathbf{B}_{m\times l} & \mathbf{O}_{m\times(n-l)}\\ \mathbf{O}_{(n-m)\times l} & \mathbf{O}_{(n-m)\times(n-l)}\end{pmatrix}.$$

So $\mathrm{Enc}\begin{pmatrix}\mathbf{A}\cdot\mathbf{B} & \mathbf{O}\\ \mathbf{O} & \mathbf{O}\end{pmatrix}$ can be obtained by passing $\mathrm{Enc}\begin{pmatrix}\mathbf{A}\\ \mathbf{O}\end{pmatrix}$ and $\mathrm{Enc}\begin{pmatrix}\mathbf{B} & \mathbf{O}\end{pmatrix}$ through the homomorphic multiplication operation, where Enc represents the encryption algorithm of our scheme.

(3) If $\max\{m,n,l\}=l$, then first fill the matrices $\mathbf{A}$ and $\mathbf{B}$ into a square matrix, that is, $\mathbf{A}'_{l\times l}=\begin{pmatrix}\mathbf{A}_{m\times n} & \mathbf{O}_{m\times(l-n)}\\ \mathbf{O}_{(l-m)\times n} & \mathbf{O}_{(l-m)\times(l-n)}\end{pmatrix}$, $\mathbf{B}'_{l\times l}=\begin{pmatrix}\mathbf{B}_{n\times l}\\ \mathbf{O}_{(l-n)\times l}\end{pmatrix}$. We have

$$\mathbf{A}'\cdot\mathbf{B}'=\begin{pmatrix}\mathbf{A} & \mathbf{O}\\ \mathbf{O} & \mathbf{O}\end{pmatrix}\cdot\begin{pmatrix}\mathbf{B}\\ \mathbf{O}\end{pmatrix}=\begin{pmatrix}\mathbf{A}\cdot\mathbf{B}_{m\times l}\\ \mathbf{O}_{(l-m)\times l}\end{pmatrix}.$$

So $\mathrm{Enc}\begin{pmatrix}\mathbf{A}\cdot\mathbf{B}\\ \mathbf{O}\end{pmatrix}$ can be obtained by passing $\mathrm{Enc}\begin{pmatrix}\mathbf{A} & \mathbf{O}\\ \mathbf{O} & \mathbf{O}\end{pmatrix}$ and $\mathrm{Enc}\begin{pmatrix}\mathbf{B}\\ \mathbf{O}\end{pmatrix}$ through the homomorphic multiplication operation, where Enc represents the encryption algorithm of our scheme.

Therefore, secure outsourced matrix multiplication based on GMS for rectangular matrices requires the user to fill the matrix before encryption. Subsequently, the filled square matrix is encrypted and uploaded to the cloud server. The recipient can then download it from the cloud. The results obtained are decrypted on the server, and then, the first m rows and the first l columns of the matrix are extracted based on the desired dimensions. Therefore, our solution can be applied to secure outsourced matrix multiplication of any dimension, i.e., $\mathbf{A}_{m\times n}\cdot\mathbf{B}_{n\times l}$.

Next, we will focus on the simulation experiment, which will be presented in detail in three parts: parameter selection, theoretical comparison, and simulation experiment. Our simulation experiment first implemented the fully homomorphic encryption scheme we constructed and then proceeded to implement secure outsourced matrix multiplication. The experiment was based on the HElib library [22] in C++, running on a personal laptop with a 2.60GHz CPU configuration of the

11th generation Intel(R) Core(TM) i5–11260 H. Finally, a detailed analysis and conclusions were provided based on the data obtained from the simulation experiment.

4.1 Parameter selection

As shown in Sect. 3.6, our system assumes that the client is honest and that the only threat comes from a curious server. Note that our scheme GMS has been proven to be indistinguishable under the chosen plaintext attack model (IND-CPA). This means that the ciphertexts of two messages are computationally indistinguishable. Since the server does not have access to the secret key and can only see the ciphertexts during the secure matrix multiplication, it does not learn anything from the ciphertexts. Thus, it is safe to conclude that our secure outsourced matrix multiplication scheme is secure against a curious server in the IND-CPA model.

Therefore, the choice of parameters can be determined based on the fully homomorphic encryption scheme that we have designed. The security parameter setting assumed by the DLWE assumption generally requires a security level of either 128 bits or 256 bits. To ensure the comparability of experimental data, we set the security parameter $\lambda = 128$. This means that the modulus q is a positive integer with a length of 128 bits. In order to minimize the impact of noise on the system and the experiment, we chose to set the value of t as a positive integer with a length of 19 bits, based on the analysis in Sect. 3.4. The parameters corresponding to the different plaintext dimensions n are shown in Table 2.

4.2 Theoretical analysis

In this section, we will primarily analyze the performance of our secure outsourced matrix multiplication. We compare our GMS-based secure matrix multiplication with the methods proposed by Jiang et al. [26], Hiromasa et al. [6], Huang et al. [4], and Zhu et al. [5]. As shown in Table 3. "#Ciphs" represents the number of ciphertexts needed to encrypt a matrix. "#HEMult" is the number of homomorphic multiplications required to perform a secure outsourced matrix multiplication. "Matrix dims" represent the applicable dimensions of the matrix. For the time complexity, let "Add" and "Mult" denote the ciphertext-ciphertext addition and ciphertext-ciphertext multiplication, respectively. Let "Cmult" denote plaintext-ciphertext multiplication. "Rot" denotes the rotation operation of a ciphertext along a row

Table 2 Parameter selection corresponding to different plaintext dimensions n

n	b	a	n	b	a
$n = 8$	$b = 2^7$	$a = 29$	$n = 128$	$b = 2^{17}$	$a = 12$
$n = 16$	$b = 2^7$	$a = 29$	$n = 256$	$b = 2^{16}$	$a = 13$
$n = 32$	$b = 2^7$	$a = 29$	$n = 512$	$b = 2^{16}$	$a = 13$
$n = 64$	$b = 2^{17}$	$a = 12$	$n = 1024$	$b = 2^{15}$	$a = 14$

Table 3 Comparison of the existing works with ours

Methods	FHE Scheme	#Ciphs	#HEMult	Matrix dims	Time complexity				
					Add	CMult	Rot	Mult	Depth
Jiang et al. [26]	BGV/CKKS	1	n	$\mathbf{A}_{m\times n} \cdot \mathbf{B}_{n\times n}$ $(m \le n)$	$6n$	$4n$	$3n + 5\sqrt{n}$	n	$2D_1 + 1D_2$
Hiromasa et al. [6]	HAO	1	1	$\mathbf{A}_{n\times n} \cdot \mathbf{B}_{n\times n}$ $\in \{0,1\}^{n\times n}$	n	–	–	$n \log n$	$1D_1 + 1D_2$
Huang et al. [4](case a)	BGV/CKKS	1	n	$\mathbf{A}_{m\times n} \cdot \mathbf{B}_{n\times l}$ $(m \le n)$	$n \cdot \log k_1 + n$	n	$n \cdot \log k_1 + n$	n	$1D_1 + 1D_2$
Huang et al. [4](case b)	BGV/CKKS	1	n	$\mathbf{A}_{m\times n} \cdot \mathbf{B}_{n\times l}$ $(m > n)$	$n \cdot \log k_1 + \log \frac{k_1}{n} + n$	n	$n \cdot \log k_1 + \log \frac{k}{n} + n$	n	$1D_1 + 1D_2$
Zhu et al. [5] (square)	BGV	1	n	$\mathbf{A}_{n\times n} \cdot \mathbf{B}_{n\times n}$	$3n$	$2n$	$4n$	n	$1D_1 + 1D_2$
Zhu et al. [5] (rectangular)	BGV	1	k_3	$\mathbf{A}_{m\times n} \cdot \mathbf{B}_{n\times l}$	$3n + \log \frac{k_1^3}{k_2 k_3^2}$	$2n$	$4n + \log \frac{k_1^3}{k_2 k_3^2}$	k_3	$1D_1 + 1D_2$
Ours (square)	GMS	1	1	$\mathbf{A}_{n\times n} \cdot \mathbf{B}_{n\times n}$	n	n	$4n$	–	$1D_1 + 1D_2$
Ours (rectangular)	GMS	1	1	$\mathbf{A}_{m\times n} \cdot \mathbf{B}_{n\times l}$	k_1	k_1	$4k_1$	–	$1D_1 + 1D_2$

(or a column), and "Depth" denotes the depth of the multiplicative circuit. D_1 and D_2 denote the circuit depths of plaintext-ciphertext multiplication and ciphertext-ciphertext multiplication, respectively. Let $k_1 = \max\{m, n, l\}$, $k_2 = \text{median}\{m, n, l\}$, $k_3 = \min\{m, n, l\}$. The specific values are provided in Table 3.

Obviously, compared to Jiang et al. [26] and Hiromasa et al. [6], our method can be applied to a wider range of matrix multiplication cases, requiring lower time complexity and circuit depth. Huang et al. [4], Zhu et al. [5] and our method can be used for matrix multiplication of arbitrary dimensions. The number of Add, Cmult, and Rot required is of the same order of magnitude, while our method only requires one homomorphic multiplication to perform a matrix multiplication. The Mult of our method is asymptotically reduced by n and k_3 times, respectively. Therefore, our GMS-based secure outsourced matrix multiplication has a lower time complexity and is more efficient compared to existing methods.

4.3 Simulation experiment

In this section, we will verify and analyze the actual performance of our scheme, through a simulation experiment. We used the parameters listed in Table 2 to implement our constructed fully homomorphic encryption scheme, GMS. In this scheme, we randomly selected and retained the plaintext **M**. We then decrypted it and compared it with the original plaintext. If they are not the same, an error will be returned. If they are the same, we recorded the encryption and decryption time, as well as the time taken for the homomorphic multiplication they were involved in. The entire program runs smoothly, and no errors are reported. Table 4 presents the size of ciphertext, the encryption and decryption times for different plaintext dimensions n, along with the time required for homomorphic multiplication with varying homomorphic multiplication times k. CipherSize denotes the size of the ciphertext. EncTime(s) represents the encryption time. DecTime(s) is the decryption time. The right side of Table 4 represents the time it takes to perform k homomorphic multiplications on the ciphertext. The data in Table 4 represent a random sample of the program run multiple times.

As shown in Table 4, the size of the ciphertext corresponding to a 64×64-dimensional plaintext is approximately only 1 MB. The encryption and decryption time

Table 4 Ciphertext size, encryption time, decryption time, and homomorphic multiplication time

n	Cipher-Size (MB)	EncTime(s)	DecTime(s)	k times homomorphic multiplication time(s)					
				1	2	3	4	5	6
8	0.09	< 0.01	< 0.01	0.11	0.20	0.31	0.42	0.56	0.68
16	0.20	0.03	0.01	0.71	1.29	2.00	2.86	3.33	4.30
32	0.79	0.07	0.06	6.01	11.94	18.00	23.99	29.98	36.01
64	1.27	0.22	0.21	39.98	75.46	114.99	152.88	188.86	225.86
128	4.99	1.22	1.10	102.03	205.96	309.20	409.81	511.46	615.15
256	18.16	12.99	11.83	836.12	1659.96	2516.55	3329.14	4177.63	5008.34

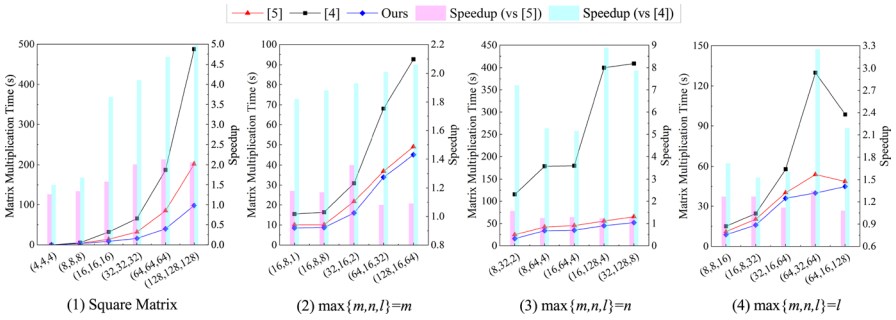

Fig. 3 The running time(s) of [4, 5], our method and speedup ([4, 5] vs our method)

for a 128×128-dimensional plaintext is only about 1 second. The homomorphic multiplication time for a 32×32-dimensional plaintext is only about 6 seconds. The fully homomorphic encryption scheme GMS that we constructed has a small ciphertext size and requires less time for encryption and decryption, resulting in high efficiency.

Next, we implemented the GMS-based secure outsourced matrix multiplication using the HElib library [22] and provided a detailed comparison of the resulting data obtained from the different secure matrix multiplication methods. Depending on the dimension of the matrix, it is divided into four cases: (1) Square matrix; (2) $\max\{m,n,l\} = m$; (3) $\max\{m,n,l\} = n$; (4) $\max\{m,n,l\} = l$. We compare the performance of our method with existing methods in Fig. 3.

From the data in Fig. 3, it is evident that in all cases, the execution time (matrix multiplication time) of our algorithm is the lowest. In the case of square matrix multiplication, our method demonstrates superior performance as it requires n times fewer homomorphic multiplications compared to other methods. Additionally, our approach requires the same number of rotations as the method proposed in [5], but significantly fewer rotations compared to the approach in [4] by a factor of $O(\log n)$ (refer to Table 3). As a result, we experience a greater speedup as the matrix dimension increases. When $(m,n,l) = (128,128,128)$, we achieve the highest speedup, up to 5X and 2X, respectively. In the case of rectangular matrix multiplication, the superiority of our approach compared to [5] is not readily apparent. This is because the matrix multiplication time of our method, [4] and [5], is positively correlated with $\max\{m,n,l\}$, n, and $\min\{m,n,l\}$, respectively. When the dimensions of the matrices are significantly different, for example, $(m,n,l) = (16,128,4)$, our method is more frequently utilized for CMult and Rot compared to the method proposed in [5]. Consequently, the time required for matrix multiplication remains essentially unchanged; however, there is a substantial growth rate compared to the results reported in [4], up to 9X.

5 Conclusion

In this paper, we propose an efficient fully homomorphic encryption scheme called GMS for matrices. The scheme has a smaller ciphertext size and less noise expansion, which allows for efficient updating of the ciphertext through key switching and modulus switching. Furthermore, GMS has significant applications in secure outsourced matrix multiplication. Through theoretical analysis and simulation experiments, our scheme demonstrates lower time complexity and superior performance compared to existing methods. It can be applied to arbitrary matrices, i.e., $\mathbf{A}_{m\times n} \cdot \mathbf{B}_{n\times l}$. Compared to existing methods, the secure outsourced matrix multiplication based on GMS has shown significant performance improvements and higher overall efficiency. Our GMS-based secure outsourced matrix multiplication is expected to be widely used in schemes that require matrix multiplication for secure outsourced computation.

In our future work, we have two goals. One goal is to enhance and extend the capacity and ability to process and handle significantly large amounts of data, thereby promoting applicability and utility in large data protection applications that utilize extensive datasets as input. The other goal is to optimize and implement multiple parallel operations to enhance the efficiency of the algorithm.

Author contributions Jianxin Gao wrote the main manuscript text, performed the main simulation experiment and prepared the figures and tables. Ying Gao reviewed and edited the manuscript, provided supervision and guidance on the research topics and provided funding and project support. All authors reviewed the manuscript.

Data availability The data is available upon request.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Zhang P, Huang T, Sun X et al (2023) Privacy-preserving and outsourced multi-party k-means clustering based on multi-key fully homomorphic encryption. IEEE Trans Dependable Secure Comput 20(3):2348–2359. https://doi.org/10.1109/tdsc.2022.3181667
2. Zhao L, Chen L (2018) Sparse matrix masking-based non-interactive verifiable (outsourced) computation, revisited. IEEE Trans Dependable Secure Comput 17(6):1188–1206. https://doi.org/10.1109/tdsc.2018.2861699
3. Duong DH, Mishra PK, Yasuda M (2016) Efficient secure matrix multiplication over lwe-based homomorphic encryption. Tatra Mt Math Publ 67(1):69–83. https://doi.org/10.1515/tmmp-2016-0031
4. Huang H, Zong H (2023) Secure matrix multiplication based on fully homomorphic encryption. J Supercomput 79(5):5064–5085. https://doi.org/10.1007/s11227-022-04850-4
5. Zhu L, Hua Q, Chen Y, et al (2023) Secure outsourced matrix multiplication with fully homomorphic encryption. In: European Symposium on Research in Computer Security, Springer, pp 249–269, https://doi.org/10.1007/978-3-031-50594-2_13

6. Hiromasa R, Abe M, Okamoto T (2016) Packing messages and optimizing bootstrapping in gsw-fhe. IEICE Trans Fundam Electron Commun Comput Sci 99(1):73–82. https://doi.org/10.1587/transfun.e99.a.73
7. Van DM, Gentry C, Halevi S, et al (2010) Fully homomorphic encryption over the integers. 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, Springer, pp 24–43
8. Regev O (2009) On lattices, learning with errors, random linear codes, and cryptography. J ACM (JACM) 56(6):1–40. https://doi.org/10.1145/1568318.1568324
9. Ĺopez-Alt A, Tromer E, Vaikuntanathan V, (2012) On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. IACR Cryptol ePrint Arch 2013:94. https://doi.org/10.1145/2213977.2214086
10. Regev O (2010) The learning with errors problem. Invit Surv CCC 7(30):11. https://doi.org/10.1109/ccc.2010.26
11. Brakerski Z, Gentry C, Vaikuntanathan V (2014) (leveled) fully homomorphic encryption without bootstrapping. ACM Trans Comput Theory (TOCT) 6(3):1–36. https://doi.org/10.1145/2633600
12. Gentry C, Sahai A, Waters B (2013) 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, Springer, pp 75–92, https://doi.org/10.1007/978-3-642-40041-4_5
13. Cheon JH, Kim A, Kim M, et al (2017) 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, Springer, pp 409–437, https://doi.org/10.1007/978-3-319-70694-8_15
14. Chillotti I, Gama N, Georgieva M et al (2020) Tfhe: fast fully homomorphic encryption over the torus. J Cryptol 33(1):34–91. https://doi.org/10.1007/s00145-019-09319-x
15. Benarroch D, Brakerski Z, Lepoint T (2017) Fhe over the integers: decomposed and batched in the post-quantum regime. In: IACR International Workshop on Public Key Cryptography, Springer, pp 271–301, https://doi.org/10.1007/978-3-662-54388-7_10
16. Canteaut A, Carpov S, Fontaine C et al (2018) Stream ciphers: a practical solution for efficient homomorphic-ciphertext compression. J Cryptol 31(3):885–916. https://doi.org/10.1007/s00145-017-9273-9
17. Genise N, Gentry C, Halevi S, et al (2019) Homomorphic encryption for finite automata. In: Advances in Cryptology—ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part II 25, Springer, pp 473–502
18. Pereira HVL (2020) Efficient agcd-based homomorphic encryption for matrix and vector arithmetic. In: International Conference on Applied Cryptography and Network Security, Springer, pp 110–129, https://doi.org/10.1007/978-3-030-57808-4_6
19. Atallah MJ, Pantazopoulos KN, Rice JR, et al (2002) Secure outsourcing of scientific computations. In: Advances in Computers, vol 54. Elsevier, pp 215–272
20. Lei X, Liao X, Huang T et al (2014) Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud. Inf Sci 280:205–217. https://doi.org/10.1016/j.ins.2014.05.014
21. Fu S, Yu Y, Xu M (2017) A secure algorithm for outsourcing matrix multiplication computation in the cloud. In: Proceedings of the Fifth ACM international workshop on security in cloud computing, pp 27–33, https://doi.org/10.1145/3055259.3055263
22. Halevi S, Shoup V (2014) 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, Springer, pp 554–571, https://doi.org/10.1007/978-3-662-44371-2_31
23. Lu W, Kawasaki S, Sakuma J (2017) Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data. In: Proceedings 2017 Network and Distributed System Security Symposium, Internet Society, https://doi.org/10.14722/ndss.2017.23119
24. Wang S, Huang H (2019) Secure outsourced computation of multiple matrix multiplication based on fully homomorphic encryption. KSII Trans Internet Inf Syst (TIIS) 13(11):5616–5630. https://doi.org/10.3837/tiis.2019.11.019

25. Lu W, Sakuma J (2018) More practical privacy-preserving machine learning as a service via efficient secure matrix multiplication. In: Proceedings of the 6th Workshop on Encrypted Computing and Applied Homomorphic Cryptography, pp 25–36, https://doi.org/10.1145/3267973.3267976
26. Jiang X, Kim M, Lauter K, et al (2018) 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, https://doi.org/10.1145/3243734.3243837
27. Micciancio D, Walter M (2017) Gaussian sampling over the integers: Efficient, generic, constant-time. In: Advances in Cryptology—CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part II 37, Springer, pp 455–485, https://doi.org/10.1007/978-3-319-63715-0_16
28. Genise N, Micciancio D, Polyakov Y (2019) Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In: Advances in Cryptology—EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part II 38, Springer, pp 655–684, https://doi.org/10.1007/978-3-030-17656-3_23
29. Katz J, Lindell Y (2020) Introduction to modern cryptography, 3rd edn. Chapman and Hall CRC, London. https://doi.org/10.1201/9781351133036