The Journal of Supercomputing (2024) 80:26394–26434
https://doi.org/10.1007/s11227-024-06428-8

Secure and efficient general matrix multiplication on cloud using homomorphic encryption

Yang Gao[1] · Gang Quan[2] · Soamar Homsi[3] · Wujie Wen[4] · Liqiang Wang[1]

Accepted: 2 August 2024 / Published online: 26 August 2024

Abstract
Despite the enormous technical and financial advantages of cloud computing, security and privacy have always been the primary concerns for adopting cloud computing facilities, especially for government agencies and commercial sectors with high-security requirements. Homomorphic encryption (HE) has recently emerged as an effective tool in ensuring privacy and security for sensitive applications by allowing computing on encrypted data. One major obstacle to employing HE-based computation, however, is its excessive computational cost, which can be orders of magnitude higher than its counterpart based on the plaintext. In this paper, we study the problem of how to reduce the HE-based computational cost for general matrix multiplication, i.e., a fundamental building block for numerous practical applications, by taking advantage of the single instruction multiple data operations supported by HE schemes. Specifically, we develop a novel element-wise algorithm for general matrix multiplication, based on which we propose two HE-based general matrix multiplication algorithms to reduce the HE computation cost. Our experimental results show that our algorithms significantly outperform the state-of-the-art approaches of HE-based matrix multiplication.

Keywords Homomorphic encryption · Privacy protection · Matrix multiplication · Cloud computing

1 Introduction

Cloud computing has become an attractive solution for industry and individuals due to its flexibility, scalability, reliability, sustainability, and affordability [1, 2]. Despite the tremendous technical and business advantages of cloud computing, security has been

Y. Gang and L. Wang were supported in part by NSF grant 1952792 and 2321572. W. Wen was supported in part by NSF grant CNS-2348733.
Soamar Homsi: Approved for Public Release on March 06, 2024. Distribution is unlimited. Case Number: 2024-0184 (original case number(s): AFRL-2024-0944).

Extended author information available on the last page of the article

one of the primary concerns for cloud users, especially for those with high-security requirements [3, 4]. Even though cloud platforms allow their users to have full control over security settings and policies, public cloud infrastructures are commonly shared among different users and applications, making the applications vulnerable to malicious attacks.

Homomorphic encryption (HE) [5–7] has emerged as an effective tool to address the security and privacy concerns while outsourcing data and computation to untrusted third parties, such as public cloud service providers. HE ensures data confidentiality during transit and processing by guaranteeing that the decrypted results are identical to the outcome when the same operations are applied to the data in plaintext. HE has raised increasing interest from many researchers and practitioners of security- and privacy-sensitive cloud applications in various domains such as health care, finance, and government agencies. One of the grand challenges, however, is how to deal with the tremendous computational cost for HE computations, which can be orders of magnitude higher than that in the plaintext space [8]. Unless HE computation cost can be effectively reduced, it would be infeasible to apply HE schemes in practical cloud applications.

In this paper, we study the problem of how to reduce HE computation cost for general matrix multiplications (MM) by taking advantage of the single instruction multiple data (SIMD) scheme for HE operations [9]. Note that different from classical SIMD technique in hardware, the SIMD scheme in the HE framework enables multiple data values to be packed into one ciphertext, and one single HE operation can be performed on all data elements in the ciphertext simultaneously. Accordingly, we develop a novel approach for HE-based MM operations, focusing on matrices in arbitrary shapes. Specifically, we make the following contributions.

1. We present a novel element-wise method for MM. This method is general and can be applied to source matrices of any shape with a significant performance improvement;
2. We develop two HE MM algorithms, with the second significantly improving upon the first. Our HE MM algorithms pack matrix elements judiciously in encrypted message "slots" and perform pertinent operations by taking advantage of the SIMD structure in HE schemes to reduce the number of primitive HE operations, such as HE multiplications, rotations, and additions, which are computationally expensive, and therefore can significantly reduce the computational cost;
3. We perform a rigorous analysis for the logical correctness of the algorithms and their complexities;
4. We implement our algorithms using a Python HE library, Pyfhel [10]. Extensive experimental results show that our proposed algorithms can significantly outperform the state-of-the-art approaches. Our code is available at https://github.com/EchizenG/HEGMM

2 Background and related work

In this section, we briefly introduce the relevant background of HE and discuss the related work.

2.1 Homomorphic encryption (HE)

Homomorphic encryption (e.g., BGV [7], BFV [11, 12], and CKKS [13]) enables computations to be performed based on encrypted data, with results still in encrypted form. As such, HE represents a promising tool to greatly enhance data privacy and security, especially when outsourcing computations to the public cloud. In the meantime, HE can be extremely computationally intensive [14], and improving its computation efficiency is key to making this technology practical for real applications.

When performing HE matrix multiplication on the cloud, source matrices are first encrypted by clients and transferred to the cloud, and the results are transferred back to clients for decryption. Encrypting each individual element of a matrix into one cyphertext can lead to excessive encryption, decryption, and communication costs, in addition to a large number of HE operations. Table 1 shows our profiling results on encryption/decryption latency, message size, and computational costs with different HE operations (More detailed experimental settings are discussed in Sec. 4.1).

To this end, Gentry and Halevi [9] proposed an efficient key generation technique that enables SIMD operations in HE. By encrypting multiple data items into one ciphertext, one single operation can be applied to all encrypted elements in the same ciphertext simultaneously, and thus, space and computing resources can be used more efficiently.

As an example, BFV [11, 12] can support a number of primitive HE operations such as HE-Add (addition), HE-Mult (multiplication), HE-CMult (constant multiplication), and HE-Rot (rotation). Given ciphertexts $ct_x = Enc(x_0, x_1, \ldots, x_n)$, $ct_y = Enc(y_0, y_1, \ldots, y_n)$ and a plaintext $pt = (p_0, p_1, \ldots, p_n)$, we have

- HE-Add: $ct_x + ct_y = Enc(x_0 + y_0, x_1 + y_1, \ldots, x_n + y_n)$
- HE-Mult: $ct_x \times ct_y = Enc(x_0 \times y_0, x_1 \times y_1, \ldots, x_n \times y_n)$
- HE-CMult: $ct_x \times pt = Enc(x_0 \times p_0, x_1 \times p_1, \ldots, x_n \times p_n)$

Table 1 Comparison of computational cost for HE v.s. Plaintext Operations. **CC** is the multiplication between two ciphertexts while **CP** is the multiplication between ciphertext and plaintext

Operations	HE	Plaintext	Ratio
Encryption (ms)	5.5	–	–
Decryption (ms)	2.57	–	–
Message size (MB)	0.5	–	–
Addition (ms)	0.550	0.009	61.1
Multiplication (ms)	20.874 (CC)	0.035	596.4
	4.138 (CP)	0.035	118.23
Rotation (ms)	5.35	0.13	41.15

- HE-Rot: $Rot(ct_x, i) = Enc(x_i, x_{i+1}, .., x_n, x_0, .., x_{i-1})$

The HE operations are computationally intensive and can consume excessive computational time. In addition, HE operations also introduce noises when performed on encrypted data [15], which must be well under control for the results to be decrypted successfully. Several HE operations, especially HE-Mult, can be extremely time-consuming (approximately 600× higher than its counterpart as shown in Table 1) and introduce much larger noise [16]. Therefore, reducing the number of HE operations (especially the HE-Mult operations) becomes critical in designing practical applications, such as matrix multiplication, under the HE framework.

2.2 Related work

There are numerous research efforts on improving the computational efficiency of MM (e.g., [17, 18, 18–22]). However, none of them can be readily adapted to optimize the computation efficiency of MM in the context of HE computation.

A naive method for HE MM is to encrypt each row/column in each matrix and then compute it using the traditional MM method. For the HE MM of $\mathcal{A}_{m\times l} \times \mathcal{B}_{l\times n}$, this would result in excessive storage requirements and computation times: $m \times n$ encrypted messages and totally $m \times l \times n$ HE-Mult operations. Another simple and intuitive approach (e.g., [23]) is to transform the MM problem into the matrix–vector multiplication problem and then adopt the SIMD scheme [9, 24] to perform the calculation. However, this requires $m + n$ ciphertexts and $m \times n$ homomorphic multiplication operations, which are still very costly.

Duong et al. [25] and Mishra et al. [26] presented approaches to packing the source matrix into a single polynomial and performing HE MM based on secure computation of the inner product of encrypted vectors. Their method works well for a single HE MM of squared matrices with well-defined dimensions. However, it is not clear how this solution can be applied to general MM. Additionally, the encrypted message packing methods used in this approach are closely related to the matrix dimensions. While HE MM of three or more matrices is shown to be possible [26], it is unclear how to perform HE matrix operations involving both multiplications and additions/subtractions without decrypting/recrypting the intermediate results, especially for matrices with arbitrary shapes.

Jiang et al. [27] proposed an intriguing HE MM approach for square matrix with $O(d)$ computational complexity. They expanded their HE MM algorithm to handle rectangle MM ($\mathcal{A}_{l\times d} \times \mathcal{B}_{d\times d}$) with $l \leq d$ and $d \mod l = 0$. However, to suit the shape requirements for matrix multiplication with variable shapes, source matrices may need to be enlarged, which can lead to increased processing time and higher resource utilization. This trade-off is important to consider when applying the algorithm to matrices that do not naturally conform to the original square or rectangular constraints.

Huang et al. [28] advocated for the use of blocking to more effectively handle matrix multiplication (MM) by treating source matrices as block matrices composed

of smaller square matrices. This approach is particularly appealing for large matrices that cannot fit into a single ciphertext, as it allows for more manageable and efficient computation. By breaking down large matrices into smaller, square blocks, the method ensures that operations remain feasible even when dealing with extensive data. However, this technique is limited in its application to large matrices that are square matrices or pairs of rectangular matrices where the number of columns and rows are integer multiples of each other. These restrictions means that while the blocking method could be highly efficient for certain configurations, it may not be as versatile when working with matrices that do not meet these specific dimensional criteria.

Rathee et al. [29] proposed to encrypt source matrices into the two-dimensional hypercube structure [24] and then transform the MM problem to a series of matrix–vector multiplication problems. Huang et al. [30] extended this approach to make it applicable to general MM. In this paper, we develop a novel element-wise MM approach. As demonstrated in Sect. 3.3, our approach can lead to a higher computational efficiency and memory usage compared to that by Huang et al. [30].

3 Our approaches

When performing HE matrix multiplication in the SIMD manner, we need to make sure that two operands are aligned and located at the same location, i.e., the same slot in the two encoded ciphertexts. Rearranging individual slots in an encrypted message can be costly. Therefore, a key to the success of reducing the computational complexity of the HE MM is how to perform the MM using element-by-element additions and multiplication operations. In what follows, we first introduce a novel algorithm to calculate MM with arbitrary dimensions using element-wise additions and multiplications. We then discuss in more detail how we implement the HE MM algorithm on packed ciphertexts in the SIMD manner and its enhanced version.

3.1 The matrix multiplication method using element-wise computations

Consider an MM problem, $\mathcal{C}_{m\times n} = \mathcal{A}_{m\times l} \times \mathcal{B}_{l\times n}$, where m, l, and $n \in \mathbb{Z}^+$. Our goal is to develop an algorithm such that $\mathcal{C}_{m\times n} = \sum_i \mathcal{A}_i \odot \mathcal{B}_i$, where $\mathcal{A}_i$, $\mathcal{B}_i$ are certain transformations of $\mathcal{A}_{m\times l}$, $\mathcal{B}_{l\times n}$, respectively, and $\odot$ represents the element-wise multiplication. For ease of our presentation, we define four matrix transformation operators as follows:

$$\sigma(\mathcal{A})_{i,j} = \mathcal{A}_{i,[i+j]_l}, \quad 0 \le i < m, 0 \le j < l \tag{1}$$

$$\tau(\mathcal{B})_{i,j} = \mathcal{B}_{[i+j]_l,j}, \quad 0 \le i < l, 0 \le j < n \tag{2}$$

$$\epsilon^k_{m\times n}(\mathcal{A})_{i,j} = \mathcal{A}_{i,[j+k]_l}, \quad 0 \le i < m, 0 \le j < n \tag{3}$$

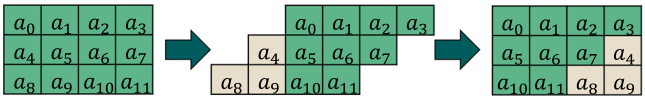
$$\omega^k_{m\times n}(\mathcal{B})_{i,j} = \mathcal{B}_{[i+k]_l,j}, \quad 0 \le i < m, 0 \le j < n \tag{4}$$

where $[x]_y$ denotes x mod y.

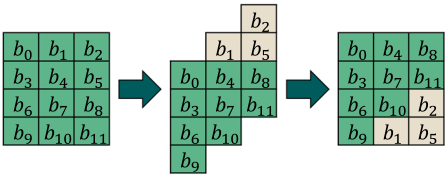
The two transformation operators, σ and τ, are similar to those introduced in [27], but more general and applicable to an arbitrary shape matrix instead of a square matrix alone. Essentially, a σ transformation rotates each row of a matrix horizontally by its corresponding row index (for example, each element in the 2_{nd} row is cyclically rotated 2 positions to the left), and a τ transformation rotates a column by its corresponding column index. Figure 1a, b illustrate examples of σ and τ transformations.

We define two new transformation operators, $\epsilon^k_{m\times n}$ and $\omega^k_{m\times n}$, with respect to a matrix of arbitrary shape. Given $\mathcal{C}_{m\times n} = \mathcal{A}_{m\times l} \times \mathcal{B}_{l\times n}$, operator $\epsilon^k_{m\times n}(\mathcal{A})$ generates a matrix with size of $m \times n$ from $\mathcal{A}_{m\times l}$ (duplicating or cropping columns when necessary), by shifting matrix $\mathcal{A}_{m\times l}$ to the left for k columns. Similarly, $\omega^k_{m\times n}(\mathcal{B})$ generates a matrix with size of $m \times n$ from $\mathcal{B}_{l\times n}$ (duplicating or cropping rows when necessary), by shifting matrix $\mathcal{B}_{l\times n}$ upward for k rows. Figure 1c, d illustrate the transformation operators $\epsilon^0_{3\times5}(\mathcal{A})$, $\epsilon^1_{3\times5}(\mathcal{A})$, $\omega^0_{3\times5}(\mathcal{B})$, and $\omega^1_{3\times5}(\mathcal{B})$, respectively.

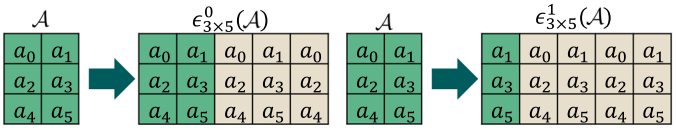
With the operators defined above, we can perform a general MM using the element-wise operations as follows:



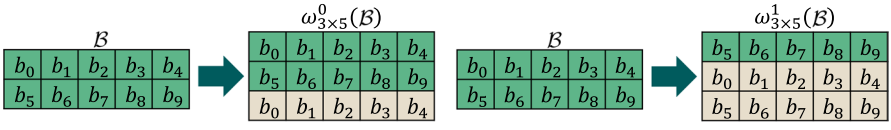
(a) σ operator: rotating i_{th} row left by i slots.



(b) τ operator: rotating j_{th} column upward by j slots.



(c) $\epsilon^0_{3\times5}(\mathcal{A}_{3\times2})$ and $\epsilon^1_{3\times5}(\mathcal{A}_{3\times2})$ with $\mathcal{C}_{3\times5} = \mathcal{A}_{3\times2} \times \mathcal{B}_{2\times5}$.



(d) $\omega^0_{3\times5}(\mathcal{B}_{2\times5})$ and $\omega^1_{3\times5}(\mathcal{B}_{2\times5})$ with $\mathcal{C}_{3\times5} = \mathcal{A}_{3\times2} \times \mathcal{B}_{2\times5}$.

Fig. 1 The illustration of σ, τ, ϵ, and ω transformation operators

$$\mathcal{A}_{m\times l} \times \mathcal{B}_{l\times n} = \sum_{k=0}^{l-1} (\epsilon^k_{m\times n} \circ \sigma(\mathcal{A})) \odot (\omega^k_{m\times n} \circ \tau(\mathcal{B})), \tag{5}$$

where $\circ$ represents the composition operation. Note that the multiplication (i.e., $\odot$) in Eq. (5) is element-wise and applied to the entire operands. Figure 2 shows an example of how an MM can be conducted based on Eq. (5). Given two source matrices, i.e., $\mathcal{A}_{5\times 3} \times \mathcal{B}_{3\times 4}$, with $m = 5, l = 3$, and $n = 4$, σ and τ transformations are conducted on $\mathcal{A}$ and $\mathcal{B}$, respectively. Then, three iterations of ϵ and ω transformations are performed to obtain three partial products, which are accumulated to get the final product. We have the following proof sketch to show that the above method indeed produces the correct MM results for arbitrary matrices.

$$\begin{aligned}
&\sum_{k=0}^{l-1} (\epsilon^k_{m\times n} \circ \sigma(\mathcal{A}))_{i,j} \cdot (\omega^k_{m\times n} \circ \tau(\mathcal{B}))_{i,j} \\
&= \sum_{k=0}^{l-1} \sigma(\mathcal{A})_{i,[j+k]_l} \cdot \tau(\mathcal{B})_{[i+k]_l,j} \\
&= \sum_{k=0}^{l-1} \mathcal{A}_{i,[i+j+k]_l} \cdot \mathcal{B}_{[i+j+k]_l,j} \\
&= \sum_{k=0}^{l-1} \mathcal{A}_{i,k} \cdot \mathcal{B}_{k,j} = (\mathcal{A} \cdot \mathcal{B})_{i,j}
\end{aligned} \tag{6}$$

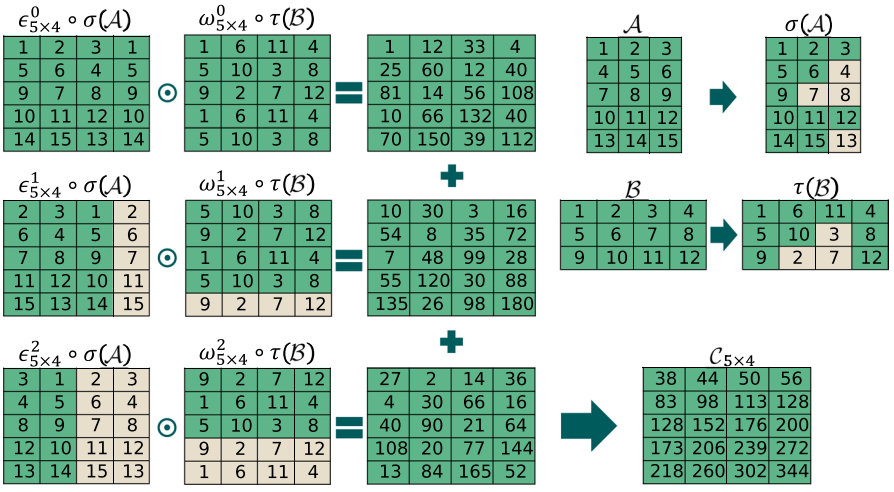


Fig. 2 An illustration example of the element-wise MM for $\mathcal{A}_{5\times3} \times \mathcal{B}_{3\times4}$ with $m = 5, l = 3$, and $n = 4$, σ and τ transformations are first conducted on $\mathcal{A}$ and $\mathcal{B}$, respectively. Then, three iterations of ϵ and ω transformations are performed to obtain three partial products, which are accumulated to get the final product

3.2 The HE-based General Matrix Multiplication (HEGMM)

With the element-wise matrix multiplication method introduced above, we are now ready to present our approach for HE matrix multiplication in the SIMD manner. As mentioned before, it is critical to minimize the number of HE operations (such as those, especially the HE-Mult, as shown in Table 1) and thus reduce the computational cost. In this subsection, we first introduce how transformations, such as $\sigma, \tau, \epsilon^k_{m\times n}$, and $\omega^k_{m\times n}$, are performed using the primitive HE operations. We then present our first algorithm for HE MM based on the element-wise matrix multiplication strategy presented above.

3.2.1 Linear transformation

To perform MM under the HE framework, two-dimensional matrices need to be flattened (either with column-major or row-major order) into one-dimensional ciphertexts, and all operations are performed on the ciphertexts. Therefore, a critical challenge to implement the computational strategy in Eq. (5) is how to efficiently conduct $\sigma, \tau, \epsilon^k_{m\times n}$, and $\omega^k_{m\times n}$, $k = \{0, \ldots, (l-1)\}$ transformation operations. Note that an arbitrary linear transformation over a vector $\boldsymbol{m}$, i.e., $L : \mathcal{R}_x \to \mathcal{R}_y$, can be expressed as $L : \boldsymbol{m} \to U \cdot \boldsymbol{m}$, where $\mathbf{U} \in \mathcal{R}_{y\times x}$ is the transformation matrix. As shown by Halevi and Shoup [24], matrix–vector multiplications can be calculated using the combination of rotation and element-wise multiplication operations. Specifically, for $0 \le z < x$, let the *z-th diagonal vector* of $\mathbf{U}$ be

$$\boldsymbol{u}_z = \begin{cases} (\underbrace{U_{0,z}, U_{1,z+1}, \ldots, U_{x-z-1,x-1}}_{|x|}, 0, \ldots, 0) & \text{if } z \ge 0 \\ (\underbrace{0, \ldots, 0, U_{z,0}, U_{z+1,1}, \ldots, U_{y-1,y-z-1}}_{|x|}) & \text{if } z < 0 \end{cases}$$

where x and y are the matrix dimensions and z is the index of diagonal vector.

Then, we have

$$\mathbf{U} \cdot \boldsymbol{m} = \sum_{-y\le z<x} (\boldsymbol{u}_z \odot \mathit{HE\text{-}Rot}(\boldsymbol{m}; z)), \tag{7}$$

where $\odot$ denotes the component-wise multiplication.

According to Eq. (7), we can construct the transformations defined in Eqs. (1)–(4) with flattened matrix $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$ such that

$$\sigma(\tilde{\mathcal{A}}) = \mathbf{U}^\sigma \cdot \tilde{\mathcal{A}}, \tag{8}$$

$$\tau(\tilde{\mathcal{B}}) = \mathbf{U}^\tau \cdot \tilde{\mathcal{B}}, \tag{9}$$

$$\epsilon^k_{m\times n}(\tilde{\mathcal{A}}) = \mathbf{U}^{\epsilon^k_{m\times n}} \cdot \tilde{\mathcal{A}}, \tag{10}$$

$$\omega^k_{m\times n}(\tilde{\mathcal{B}}) = \mathbf{U}^{\omega^k_{m\times n}} \cdot \tilde{\mathcal{B}}. \tag{11}$$

Let $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$ be source matrices flattened in the column-major order. By generalizing the location change patterns for the operators, we can define $\mathbf{U}^{\sigma}$, $\mathbf{U}^{\tau}$, $\mathbf{U}^{\epsilon^k_{m\times n}}$ and $\mathbf{U}^{\omega^k_{m\times n}}$ as follows:

$$\mathbf{U}^{\sigma}_{i+j\cdot m,h} = \begin{cases} 1 & \text{if } h = i + [i+j]_l \cdot m, \\ 0 & \text{otherwise;} \end{cases} \tag{12}$$

$$\mathbf{U}^{\tau}_{i+j\cdot l,h} = \begin{cases} 1 & \text{if } h = [i+j]_l + j \cdot l, \\ 0 & \text{otherwise;} \end{cases} \tag{13}$$

$$\mathbf{U}^{\epsilon^k_{m\times n}}_{i,j} = \begin{cases} 1 & \text{if } j = [k \cdot m + i]_{m\cdot l} \\ 0 & \text{otherwise;} \end{cases} \tag{14}$$

$$\mathbf{U}^{\omega^k_{m\times n}}_{i,j} = \begin{cases} 1 & \text{if } j = [k + [i]_m]_l + \lfloor i/m \rfloor \cdot l \\ 0 & \text{otherwise;} \end{cases} \tag{15}$$

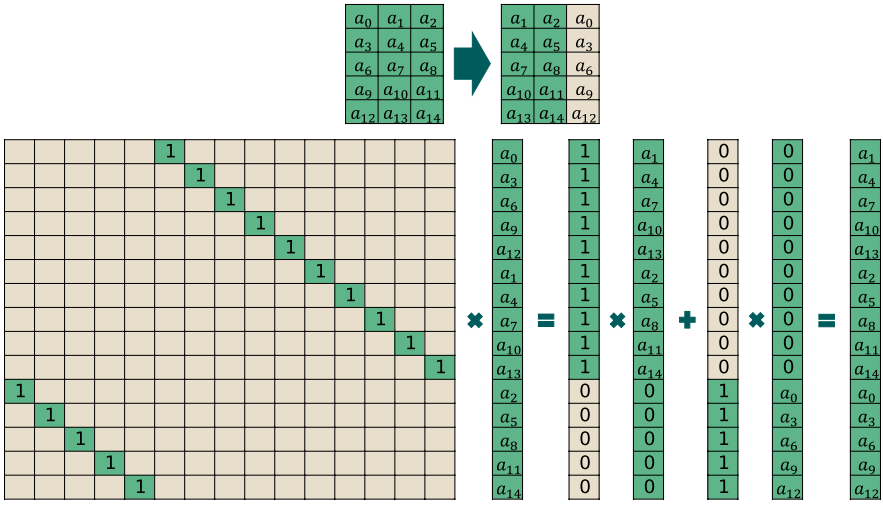
For the sake of clarity, scopes of i, j and h in Eqs. (12)–(15) are listed below.

	i	j	h
$\mathbf{U}^{\sigma}$	$[0, m)$	$[0, l)$	$[0, ml)$
$\mathbf{U}^{\tau}$	$[0, l)$	$[0, n)$	$[0, nl)$
$\mathbf{U}^{\epsilon^k_{m\times n}}$	$[0, mn)$	$[0, ml)$	N/A
$\mathbf{U}^{\omega^k_{m\times n}}$	$[0, mn)$	$[0, nl)$	N/A

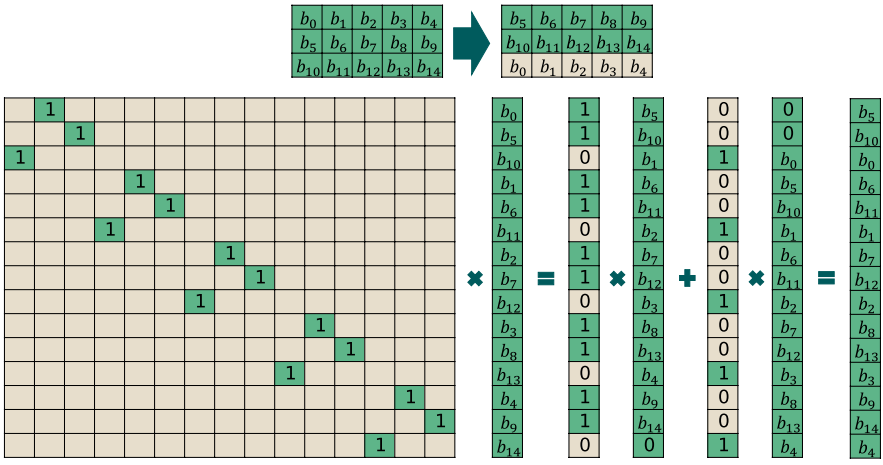
When $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$ are matrices flattened in the row-major order, similar transformation matrices can be constructed. We omit it due to page limit.

Note that, from Eq. (7), the σ, τ, $\epsilon^k_{m\times n}$, and $\omega^k_{m\times n}$ operations can be realized using a sequence of HE-Rot, HE-CMult, and HE-Add operations. Figure 3 shows examples of transformations $\epsilon^1_{5\times 3}(\mathcal{A})$ and $\omega^1_{3\times 5}(\mathcal{B})$ as well as the associated matrices $\mathbf{U}^{\epsilon^1_{5\times 3}}$ and $\mathbf{U}^{\omega^1_{3\times 5}}$ for matrix $\mathcal{A}_{5\times 3}$ and $\mathcal{B}_{3\times 5}$, respectively. In the meantime, Eq. (7) clearly shows that the computational cost depends heavily on how many diagonal vectors (i.e., $\boldsymbol{u}_z$ in Eq. (7)) in the corresponding transformation matrices, i.e., $\mathbf{U}^{\sigma}$, $\mathbf{U}^{\tau}$, $\mathbf{U}^{\epsilon^k_{m\times n}}$, and $\mathbf{U}^{\omega^k_{m\times n}}$, are nonzeros. The more the nonzero diagonal vectors are, the higher the computation costs become. To this end, we have the following theorems that reveal important properties related to nonzero diagonal vectors in these transformation matrices.

Theorem 3.1 *Let $\sigma(\mathcal{A}) = \mathbf{U}^{\sigma}\mathcal{A}$ for $\mathcal{A}$ with a dimension of $m \times l$. There are at most $2 \cdot \min(m, l) - 1$ nonzero diagonals in $\mathbf{U}^{\sigma}$ no matter whether the matrix is flattened with a column-major or row-major order.*



(a) The process of linear transformation for $\epsilon^1_{5\times3}$ transformation on matrix $\mathcal{A}_{5\times3}$. $\boldsymbol{u}_5$ is the vector with 10 1's and 5 0's while $\boldsymbol{u}_{10}$ with 10 0's and 5 1's. Therefore, according to Equation 7, it rotates flattened $\bar{\mathcal{A}}$ 5 slots and times $\boldsymbol{u}_5$. Then it rotates flattened $\bar{\mathcal{A}}$ 10 slots reversely and times $\boldsymbol{u}_{10}$. Finally, add all above partial products together.



(b) The process of linear transformation for $\omega^1_{3\times5}$ transformation on matrix $\mathcal{B}_{3\times5}$. $\boldsymbol{u}_1$ is the vector with ten 1's and five 0's while $\boldsymbol{u}_2$ with 10 0's and 5 1's. Therefore, according to Equation 7, it rotates flattened $\bar{\mathcal{B}}$ 1 slots and times $\boldsymbol{u}_1$. Then it rotates flattened $\bar{\mathcal{A}}$ 2 slots reversely and times $\boldsymbol{u}_2$. Finally, add all above partial products together.

Fig. 3 The permutation matrices $\mathbf{U}^{\epsilon^1_{5\times3}}$ and $\mathbf{U}^{\omega^1_{3\times5}}$ and linear transformations of $\epsilon^1_{5\times3}(\mathcal{A}_{5\times3})$ and $\omega^1_{3\times5}(\mathcal{B}_{5\times3})$

Theorem 3.2 *Let $\tau(\mathcal{B}) = \mathbf{U}^{\tau}\mathcal{B}$ for $\mathcal{B}$ with a dimension of $l \times n$. There are at most $2 \cdot \min(n, l) - 1$ nonzero diagonals in $\mathbf{U}^{\tau}$ no matter if the matrix is flattened with a column-major or row-major order.*

Theorem 3.3 *Let $\epsilon^k_{m\times n}(\mathcal{A}) = \mathbf{U}^{\epsilon^k_{m\times n}}\mathcal{A}$ be the linear transformation $\epsilon_{m\times n} : \mathcal{R}_{m\times l} \to \mathcal{R}_{m\times n}$ with matrix $\mathcal{A}$ having a dimension of $m \times l$. There are at most $\left\lfloor \frac{n}{l} \right\rfloor + 1$ nonzero diagonal vectors in $\mathbf{U}^{\epsilon^k_{m\times n}}$ when the matrix is flattened with the column-major order. There are at most $(\left\lfloor \frac{n}{l} \right\rfloor + 2) \cdot m$ nonzero diagonal vectors in $\mathbf{U}^{\epsilon^k_{m\times n}}$ when matrix $\mathcal{A}$ is flattened with the row-major order. Specifically, when $n = l$, there are no more than two nonzero diagonals in $\mathbf{U}^{\epsilon^k_{m\times n}}$, no matter if the matrix is flattened in column-major or row-major order.*

Theorem 3.4 *Let $\omega^k_{m\times n}(\mathcal{B}) = \mathbf{U}^{\omega^k_{m\times n}}\mathcal{B}$ be the linear transformation $\omega_{m\times n} : \mathcal{R}_{l\times n} \to \mathcal{R}_{m\times n}$ with matrix $\mathcal{B}$ having a dimension of $l \times n$. There are at most $(\left\lfloor \frac{m}{l} \right\rfloor + 2) \cdot n$ nonzero diagonal vectors in $\mathbf{U}^{\omega^k_{m\times n}}$ when the matrix is flattened with column-major order. There are at most $\left\lfloor \frac{m}{l} \right\rfloor + 1$ nonzero diagonal vectors in $\mathbf{U}^{\omega^k_{m\times n}}$ when matrix $\mathcal{B}$ is flattened with row-major order. Specifically, when $m = l$, there are no more than two nonzero diagonals in $\mathbf{U}^{\omega^k_{m\times n}}$, no matter if the matrix is flattened in column-major or row-major order.*

The proofs for Theorems 3.1–3.4 can be found in Appendix 1.

According to Theorems 3.1 and 3.2, the numbers of nonzero diagonal vectors in $\mathbf{U}^{\sigma}$ and $\mathbf{U}^{\tau}$ depend solely on the dimensions of corresponding matrices and are independent of how the matrices are flattened. However, as shown in Theorems 3.3 and 3.4, the numbers of nonzero diagonal vectors in $\mathbf{U}^{\epsilon^k_{m\times n}}$ and $\mathbf{U}^{\omega^k_{m\times n}}$ depend on not only the dimensions of matrices but also the way they are flattened.

3.2.2 The HEGMM algorithm

HEGMM is a straightforward implementation of Eq. (5). We first conduct σ and τ transformations (lines 2–3) on the source matrices of $\mathcal{A}$ and $\mathcal{B}$. We then go through a loop (lines 5–9) that apply $\epsilon^k_{m\times n}$ and $\tau^k_{m\times n}$ transformations and element-wise multiplication and addition to calculate and accumulate the partial product. The final result can be obtained by decrypting the sum of the product (line 11).

Table 2 summarizes the time complexity of HEGMM for each step. The computational complexity of Algorithm 1 mainly comes from the required HE operations associated with the $\sigma, \tau, \epsilon^k_{m\times n}$, and $\omega^k_{m\times n}$ operations. Assuming $\mathcal{A}$ and $\mathcal{B}$ are encrypted, from Theorem 3.1 and Theorem 3.2, we know that there are $2\min(m, l) - 1$ (resp. $2\min(n, l) - 1$) non-diagonals for the σ (resp. τ) operation. Therefore, according to Eq. (7), the σ (resp. τ) operation requires $2\min(m, l) - 1$ (resp. $2\min(n, l) - 1$) HE-CMult, HR-Rot, and HR-Add operations. These computational costs have nothing to do with how the matrices are flattened (e.g., in

column-major or in row-major order), and they also become trivial if they are performed on $\mathcal{A}$ and $\mathcal{B}$ in plaintext.

According to theorem 3.3 and theorem 3.4, one ϵ and ω transformation needs $\left\lfloor \frac{n}{l} \right\rfloor + 1$ and $\left\lfloor \frac{m}{l} \right\rfloor + 2 \cdot n$ HE-Add, HE-CMult, and HE-Rot operations, respectively. It is not difficult to see from the algorithm that HEGMM has a HE multiplication depth of **1 M+2C** with one HE multiplication (e.g., **1 M**) and two constant multiplication (e.g., **2C**).

Note that the ϵ and ω transformation, which must be performed multiple times in the cloud, require HE operations depending on not only the dimensions of matrices but also the way they are flattened. As a result, the computational complexities can be dramatically different under different scenarios, as shown in Theorems 3.3 and 3.4. In the next subsection, we show how we can take advantage of this fact and reduce the computation cost.

Algorithm 1 HEGMM: HE-based General Matrix Multiplication

Input: matrix $\mathcal{A}_{m\times l}$ and matrix $\mathcal{B}_{l\times n}$
Output: $\mathcal{C}_{m\times n}$
[Step1]
$ct.\mathcal{A}^{(0)} \leftarrow \sigma(\mathcal{A})$
$ct.\mathcal{B}^{(0)} \leftarrow \tau(\mathcal{B})$
[Step2]
for $k = 0$ to $l - 1$ **do**
  $ct.\mathcal{A}^{(k)} \leftarrow \epsilon^{k}_{m\times n}(ct.\mathcal{A}^{(0)})$
  $ct.\mathcal{B}^{(k)} \leftarrow \omega^{k}_{m\times n}(ct.\mathcal{B}^{(0)})$
  $ct.\mathcal{C} \leftarrow ct.\mathcal{C} + ct.\mathcal{A}^{(k)} \odot ct.\mathcal{B}^{(k)}$
end
[Step3]
$\mathcal{C}_{m\times n} \leftarrow ct.\mathcal{C}$
return $\mathcal{C}_{m\times n}$

Table 2 Time complexity of HEGMM where $\mu = \min(m, l)$, $\nu = \min(n, l)$, $\mathcal{E} : \left\lfloor \frac{n}{l} \right\rfloor + 1$, $\mathcal{W} : (\left\lfloor \frac{m}{l} \right\rfloor + 2\cdot)n$

Line	HE-Add	HE-Mult	He-CMult	HE-Rot	Depth
2	$2\mu - 1$	0	$2\mu - 1$	$2\mu - 1$	$1C$
3	$2\nu - 1$	0	$2\nu - 1$	$2\nu - 1$	–
6	$\mathcal{E}l$	0	$\mathcal{E}l$	$\mathcal{E}l$	$1C$
7	$\mathcal{W}l$	0	$\mathcal{W}l$	$\mathcal{W}l$	–
8	l	l	0	0	$1M$
Total	$\mathcal{E}l + \mathcal{W}l$ $+2(\mu + \nu) + l$	l	$\mathcal{E}l + \mathcal{W}l$ $+2(\mu + \nu)$	$\mathcal{E}l + \mathcal{W}l$ $+2(\mu + \nu)$	$1M + 2C$

3.3 The enhanced HEGMM algorithm

In this section, we introduce a more elaborated approach for HEGMM that can be more computationally efficient. The fundamental principle we rely on to develop this algorithm is presented in Theorem 3.3 and 3.4. For the HE matrix multiplication of $\mathcal{A}_{m\times l} \times \mathcal{B}_{l\times n}$, the proposed new algorithm can significantly improve the computation efficiency when $m = \min\{m, l, n\}$ and/or $n = \min\{m, l, n\}$. If not, we can always resort to Algorithm 1 to find the solution. Therefore, in what follows, we first discuss the new approach based on two cases: (i) $m = \min\{m, l, n\}$; and (ii) $n = \min\{m, l, n\}$. We then present the algorithm and related discussions in detail.

3.3.1 Case 1: $m = \min\{m, l, n\}$

For the HE MM of $\mathcal{A}_{m\times l} \times \mathcal{B}_{l\times n}$, Algorithm 1 needs to perform l iterations, with each iteration including one ϵ transformation, one ω transformation, one HE-Add, and one HE-Mult operation. Assuming the matrix is flattened with the column-major order, according to Theorem 3.3 and 3.4, one ϵ transformation and one ω transformation would result in no more than $\left(\left(\left\lfloor\frac{n}{l}\right\rfloor + 1\right) + 2n\right)$ nonzero diagonals in corresponding transformation matrices, with each nonzero diagonal requiring one HE-Add, one HE-Rot, and one HE-CMult operations. However, if we can expand matrix $\mathcal{A}_{m\times l}$ to $\bar{\mathcal{A}}_{l\times l}$, then the number of nonzero diagonals becomes no more than $\left(\left(\left\lfloor\frac{n}{l}\right\rfloor + 1\right) + 2\right)$ instead. Since $n \geq 1$ and

$$\left(\left(\left\lfloor\frac{n}{l}\right\rfloor + 1\right) + 2n\right) \geq \left(\left(\left\lfloor\frac{n}{l}\right\rfloor + 1\right) + 2\right),$$

the number of nonzero diagonals and, thus, the computational cost can be dramatically reduced.

Note that, if we assume the matrix is flattened with the row-major order, one ϵ transformation and one ω transformation would result in no more than $\left(\left(\left\lfloor\frac{n}{l}\right\rfloor + 2\right)m + 1\right)$ nonzero diagonals in corresponding transformation matrices. When expanding matrix $\mathcal{A}_{m\times l}$ to $\bar{\mathcal{A}}_{l\times l}$, the total number of nonzero diagonals in the corresponding transformation matrices becomes $\left(\left(\left\lfloor\frac{n}{l}\right\rfloor + 2\right)l + 1\right)$, according to Theorems 3.3 and 3.4. It becomes obvious that using the column-major order is a better choice than using the row-major order in this case, since when $m > 1$, we have

$$\left(\left(\left\lfloor\frac{n}{l}\right\rfloor + 2\right)m + 1\right) \geq \left(\left(\left\lfloor\frac{n}{l}\right\rfloor + 1\right) + 2\right).$$

The question now becomes how to expand $\mathcal{A}_{m\times l}$ to $\bar{\mathcal{A}}_{l\times l}$ and maintain the logical correctness of the MM result. One intuitive approach is to expand $\mathcal{A}_{m\times l}$ by filling zeroes to the newly added elements, i.e., the elements in rows from row m to $(l - m - 1)$. The final product, as a sub-matrix, can be extracted from the product of $\bar{\mathcal{A}}_{l\times l} \times \mathcal{B}_{l\times n}$ easily. Note that expanding the matrix dimension does not increase the computational complexity in the SIMD scheme as long as the result matrix can fit in one message.

Rather than simply filling zeroes, we can expand $\mathcal{A}_{m\times l}$ by duplicating the rows of $\mathcal{A}_{m\times l}$ repeatedly. This helps to reduce the number of iterations (lines 5–9) in Algorithm 1, thanks to our observations that are formulated in the following theorem.

Theorem 3.5 *Let $\mathcal{A}_{m\times l}$ and $\mathcal{B}_{l\times n}$ with $m < l$, and let $\bar{A}$ be matrix expanded with $t = \left\lceil \frac{l}{m} \right\rceil$ copies of $\mathcal{A}$ vertically, i.e., $\bar{\mathcal{A}} = \{\bar{A}_0;\bar{A}_1;\ldots;\bar{A}_{(t-1)}\}^T$ with $\bar{A}_0 = \bar{A}_1 = \cdots = \bar{A}_{(t-1)} = \mathcal{A}_{m\times l}$. Then,*

- $\epsilon^k_{tm\times n}(\sigma(\bar{\mathcal{A}})) \odot \omega^k_{tm\times n}(\tau(\mathcal{B}))$ contains t items of $\epsilon^p_{m\times n}(\sigma(\mathcal{A})) \odot \omega^p_{m\times n}(\tau(\mathcal{B}))$, with $p \in \{[k]_l, [k+m]_l, \ldots, [k+(t-1)m]_l\}$.
- $\epsilon^k_{tm\times n}(\sigma(\bar{\mathcal{A}})) \odot \omega^k_{tm\times n}(\tau(\mathcal{B}))$, $k = 0, 1, \ldots, (m-1)$ contains all items of $\epsilon^p_{m\times n}(\sigma(\mathcal{A})) \odot \omega^p_{m\times n}(\tau(\mathcal{B}))$, with $p \in \{0, 1, \ldots, (l-1)\}$.

According to Theorem 3.5, after expanding $\mathcal{A}_{m\times l}$ with t copies of $\mathcal{A}_{m\times l}$ vertically to form $\bar{\mathcal{A}}_{tm\times l}$, each iteration of Algorithm 1 can now produce t partial products $\epsilon^p_{m\times n}(\mathcal{A}) \odot \omega^p_{m\times n}(\mathcal{B})$. As a result, the required HE computations can be greatly reduced, which can be better illustrated using the example in Fig. 4.

Figure 4 shows two source matrices $\mathcal{A}_{2\times 5}$ and $\mathcal{B}_{5\times 7}$, with $m = 2$, $l = 5$, and $n = 7$. $\bar{\mathcal{A}}$ is the matrix by duplicating $\mathcal{A}$ three times, i.e., $t = \lceil 5/2 \rceil = 3$. Note that, each $\epsilon_{6\times 7}(\sigma(\bar{\mathcal{A}})) \odot \omega_{6\times 7}(\tau(\mathcal{B}))$ contains three copies of $\epsilon_{2\times 7}(\sigma(\mathcal{A})) \odot \omega_{2\times 7}(\tau(\mathcal{B}))$, as shown in the figure: $\epsilon^0_{6\times 7}(\sigma(\bar{\mathcal{A}})) \odot \omega^0_{6\times 7}(\tau(\mathcal{B}))$ contains $\epsilon^0_{2\times 7}(\sigma(\mathcal{A})) \odot \omega^0_{2\times 7}(\tau(\mathcal{B}))$, $\epsilon^2_{2\times 7}(\sigma(\mathcal{A})) \odot \omega^2_{2\times 7}(\tau(\mathcal{B}))$, and $\epsilon^4_{2\times 7}(\sigma(\mathcal{A}) \odot \omega^4_{2\times 7}(\tau(\mathcal{B}))$. We then need to add all the partial products together to get the final result.

As such, by duplicating $\mathcal{A}_{m\times l}$ into $\bar{\mathcal{A}}_{tm\times l}$, we can reduce not only the HE operations associated with the ϵ and ω operations but also the HE-Mult operations (i.e., at most m HE-Mult operations according to Theorem 3.5) for partial production calculations, which is highly costly. Even though extra HE rotations are needed to extract the partial results, the computation cost is much smaller than that of HE-Mult as shown in Table 1. It is worth mentioning that, while one $\epsilon^k_{m\times n}(\bar{\mathcal{A}}) \odot \omega^k_{m\times n}(\mathcal{B})$ helps to produce multiple copies of $\epsilon^p_{m\times n}(\mathcal{A}) \odot \omega^p_{m\times n}(\mathcal{B})$, as shown in Fig. 4, some of them may be produced repeatedly. These redundant copies should be identified, which can be easily identified according to Theorem 3.5, and excluded from the final results.

3.3.2 Case 2: $n = \min\{m, l, n\}$

We can employ the same analysis flow as above. There are two major differences compared with the case of $m = \min\{m, l, n\}$. (i) We duplicate matrix $\mathcal{B}$ horizontally to expand $\mathcal{B}$ instead of $\mathcal{A}$; (ii) The row-major order is a better option than the column-major order in this case.

When $n = \min\{m, l, n\}$, if the matrix is flattened with the row-major order, according to Theorem 3.3 and 3.4, one ϵ transformation and one ω transformation would result in no more than $(2m + \left\lfloor \frac{m}{l} \right\rfloor + 1)$ nonzero diagonals in corresponding transformation matrices. When expanding $\mathcal{B}_{l\times n}$ to $\bar{\mathcal{B}}_{l\times l}$, the total number of nonzero

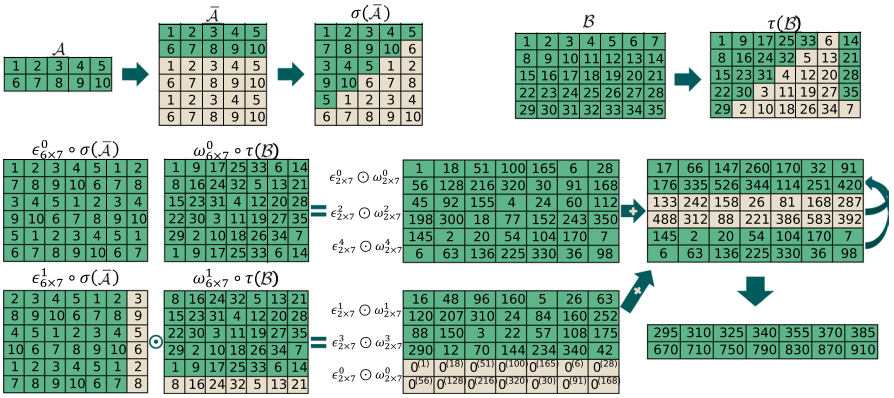


Fig. 4 An illustration example of the Enhanced HEGMM Algorithm for multiplying two matrices $\mathcal{A}_{2\times5}$ and $\mathcal{B}_{5\times7}$, with $m = 2$, $l = 5$, and $n = 7$. $\bar{\mathcal{A}}$ is the matrix by duplicating $\mathcal{A}$ 3 times, i.e., $t = \lceil 5/2 \rceil = 3$ and $\mathcal{B}_{5\times7}$ remains unchanged. The partial products are accumulated to obtain the final product. Note $\epsilon^0_{2\times7}(\sigma(\mathcal{A})) \odot \omega^0_{2\times7}(\tau(\mathcal{B}))$ is generated twice, and the duplicated partial products should be excluded from the final results

diagonals in corresponding transformation matrices is reduced to $(2 + \left\lfloor \frac{m}{l} \right\rfloor + 1)$ instead. However, if the column-major order is used, the total number of nonzero diagonals after expanding is $(1 + (2 + \left\lfloor \frac{m}{l} \right\rfloor) \cdot n)$, which makes the row-major order a better option to flatten the matrices.

Similarly, when we expand $\mathcal{B}$ by duplicating $\mathcal{B}_{l\times n}$, we can generate multiple partial products, i.e., $\epsilon^p_{m\times n}(\mathcal{A}) \odot \omega^p_{m\times n}(\mathcal{B})$, using one HE-Mult operation, as supported by the following theorem. The proof is quite similar to that for Theorem 3.5 and thus omitted due to page limit.

Theorem 3.6 *Let $\mathcal{A}_{m\times l}$ and $\mathcal{B}_{l\times n}$ with $n < l$, and let $\bar{\mathcal{B}}$ be matrix expanded with $t = \left\lceil \frac{l}{n} \right\rceil$ copies of $\mathcal{B}$ horizontally, i.e., $\bar{\mathcal{B}} = \{\mathcal{B};\mathcal{B};\ldots;\mathcal{B}\}$. Then,*

- $\epsilon^k_{m\times tn}(\sigma(\mathcal{A})) \odot \omega^k_{m\times tn}(\tau(\bar{\mathcal{B}}))$ contains t items of $\epsilon^p_{m\times n}(\sigma(\mathcal{A})) \odot \omega^p_{m\times n}(\tau(\mathcal{B}))$, with $p = [k]_l, [k+n]_l, \ldots, [k+(t-1)n]_l$;
- $\epsilon^k_{m\times tn}(\sigma(\mathcal{A})) \odot \omega^k_{m\times tn}(\tau(\bar{\mathcal{B}}), \quad k = 0, 1, \ldots, (n-1)$ contains all items of $\epsilon^p_{m\times n}(\sigma(\mathcal{A})) \odot \omega^p_{m\times n}(\tau(\mathcal{B}))$, with $p = 0, 1, \ldots, (l-1)$.

Figure 5 shows an illustrative example of HE MM with two source matrices $\mathcal{A}_{5\times4}$ and $\mathcal{B}_{4\times2}$, with $m = 5$, $l = 4$, and $n = 2$. $\bar{\mathcal{B}}$ is the matrix by duplicating $\mathcal{B}$ horizontally for two times, i.e., $t = \lceil 4/2 \rceil = 2$. Note that, each $\epsilon_{5\times4}(\sigma(\mathcal{A})) \odot \omega_{5\times4}(\tau(\bar{\mathcal{B}}))$ using one HE-Mult operation can produce two copies of $\epsilon_{5\times2}(\sigma(\mathcal{A})) \odot \omega_{5\times2}(\tau(\mathcal{B}))$, as shown in the figure: $\epsilon^0_{5\times4}(\sigma(\mathcal{A})) \odot \omega^0_{5\times4}(\tau(\bar{\mathcal{B}}))$ contains $\epsilon^0_{5\times2}(\sigma(\mathcal{A})) \odot \omega^0_{5\times2}(\tau(\mathcal{B}))$ and $\epsilon^2_{5\times2}(\sigma(\mathcal{A})) \odot \omega^2_{5\times2}(\tau(\mathcal{B}))$. $\epsilon^1_{5\times4}(\sigma(\mathcal{A})) \odot \omega^1_{5\times4}(\sigma(\bar{\mathcal{B}}))$ contains $\epsilon^1_{5\times2}(\sigma(\mathcal{A})) \odot \omega^1_{5\times2}(\tau(\mathcal{B}))$ and $\epsilon^3_{5\times2}(\sigma(\mathcal{A})) \odot \omega^3_{5\times2}(\tau(\mathcal{B}))$. We then need to

add all the partial products together to get the final result. As such, we only need to perform at most n HE-Mult operations according to Theorem 3.6 to obtain all the partial products. Redundant copies may also be generated during this process, which should be identified according to Theorem 3.6 and excluded from the final results.

Algorithm 2 HEGMM-Enhanced

Input: matrix $\mathcal{A}_{m\times l}$, $\mathcal{B}_{l\times n}$
Output: matrix $\mathcal{C}_{m\times n} = \mathcal{A}_{m\times l} \times \mathcal{B}_{l\times n}$
$p \leftarrow \min(m, l, n)$
$t \leftarrow \lceil l/p \rceil$
// Determine M and N by shape
if p=m **then**
  $\bar{\mathcal{A}}_{M\times l} \leftarrow [\underbrace{\mathcal{A}; \mathcal{A}; \mathcal{A}; ...; \mathcal{A}}_{t}]^T$
  $\bar{\mathcal{B}}_{l\times N} \leftarrow \mathcal{B}$
  $M = t \times m, N = n$
else if p=n **then**
  $\bar{\mathcal{A}}_{M\times l} \leftarrow \mathcal{A}$
  $\bar{\mathcal{B}}_{l\times N} \leftarrow [\underbrace{\mathcal{B}; \mathcal{B}; \mathcal{B}; ...; \mathcal{B}}_{t}]$
  $M = m, N = t \times n$
else
  $\bar{\mathcal{A}}_{M\times l} \leftarrow \mathcal{A}$
  $\bar{\mathcal{B}}_{l\times N} \leftarrow \mathcal{B}$
  $M = m, N = n$
end
$ct.s\bar{\mathcal{A}} \leftarrow \epsilon^0_{M\times \max(l,N)}(\sigma(\bar{\mathcal{A}}))$
$ct.t\bar{\mathcal{B}} \leftarrow \omega^0_{\max(l,M)\times N}(\tau(\bar{\mathcal{B}}))$
// Multiplication on Cloud
$ct.\mathcal{C}_{m\times n} \leftarrow ct.s\bar{\mathcal{A}} \odot ct.t\bar{\mathcal{B}}$
for $k = 0, 1, ..., (p-1)$ **do**
  $ct.\bar{\mathcal{A}}^{(k)} \leftarrow \epsilon^k_{M\times N}(ct.s\bar{\mathcal{A}})$
  $ct.\bar{\mathcal{B}}^{(k)} \leftarrow \omega^k_{M\times N}(ct.t\bar{\mathcal{B}})$
  $ct.\mathcal{C}_{temp} \leftarrow ct.\bar{\mathcal{A}}^{(k)} \odot ct.\bar{\mathcal{B}}^{(k)}$
  // $ct.\mathcal{C}_{temp}$ contains of t items of $\epsilon^{k+i\cdot p}_{m\times n}(\sigma(\mathcal{A})) \odot \omega^{k+i\cdot p}_{m\times n}(\tau(\mathcal{B}))(0 \le i < t)$.
  for $i = 0, 1, ..., (t-1)$ **do**
    $j = [k + i \cdot p]_l$
    if $\epsilon^j_{m\times n}(\sigma(\mathcal{A})) \odot \omega^j_{m\times n}(\tau(\mathcal{B})) \in ct.\mathcal{C}_{temp}$ has not been accumulated in $ct.\mathcal{C}_{m\times n}$ **then**
      $ct.\mathcal{C}_{m\times n} \leftarrow ct.\mathcal{C}_{m\times n} + \epsilon^j_{m\times n}(\sigma(\mathcal{A})) \odot \omega^j_{m\times n}(\tau(\mathcal{B}))$
  end
end
// Return encrypted result to client
return $ct.\mathcal{C}_{m\times n}$

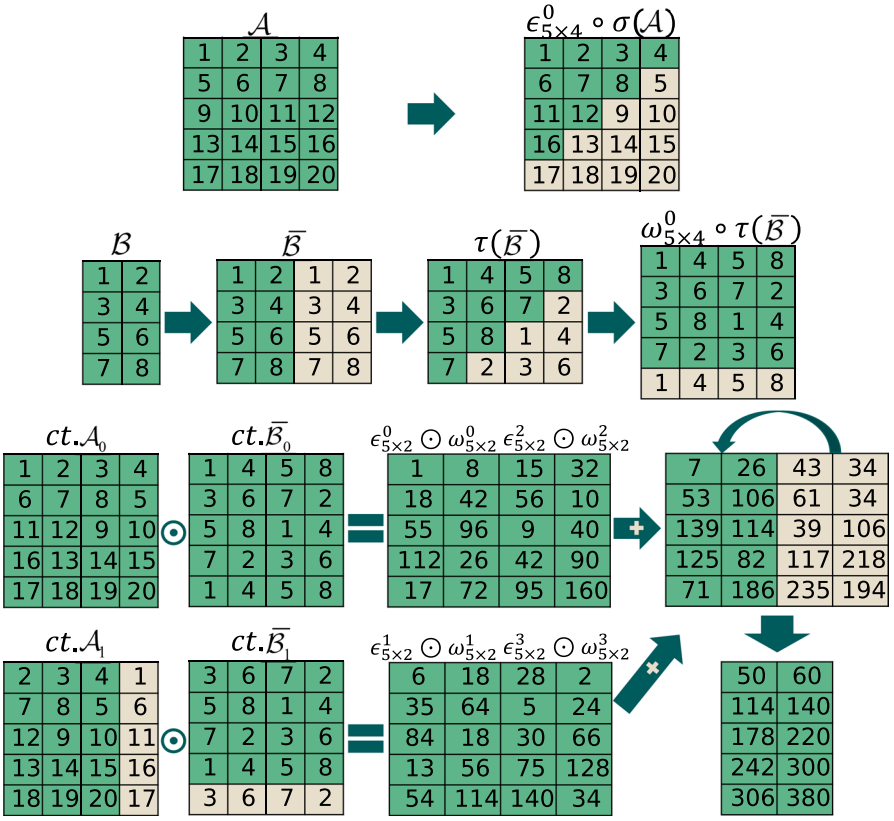


Fig. 5 This is an illustrative example of the enhanced HE MM algorithm for multiplying two matrices $\mathcal{A}_{5\times4}$ and $\mathcal{B}_{4\times2}$, with $m = 5$, $l = 4$, and $n = 2$. $\bar{\mathcal{B}}$ is the matrix by duplicating $\mathcal{B}$ horizontally for two times, i.e., $t = \lceil 4/2 \rceil = 2$ and $\mathcal{A}_{5\times4}$ remains unchanged. The partial products are accumulated to obtain the final product

The overall algorithm for the enhanced HE-based General MM, named HEGMM-En, is presented in Algorithm 2. Note that, when $m < l$ and $n < l$, we can choose to duplicate either $\mathcal{A}$ or $\mathcal{B}$. In Algorithm 2, we choose the smaller of m and n and expand either $\mathcal{A}$ or $\mathcal{B}$ accordingly (lines 3–10). When $l = \min\{m, l, n\}$, we make no change of $\mathcal{A}$ and $\mathcal{B}$ (lines 11–15). After initializing several relevant variables (lines 16–18), Algorithm 2 goes through a loop to compute and accumulate the partial products (lines 19–28). To be more specific, we first conduct ϵ and ω transformations based on the expanded matrix ($\mathcal{A}$ or $\mathcal{B}$) (lines 20–21), which are combined together into $\mathcal{C}_{temp}$ using the element-wise HE multiplication (line 22). The algorithm then extracts the possible t copies of $\epsilon_{m\times n}(\sigma(\mathcal{A})) \odot \omega_{m\times n}(\tau(\mathcal{B}))$ from $\mathcal{C}_{temp}$ and accumulates them to $\mathcal{C}_{m\times n}$, according to Theorems 3.5 and 3.6, and the redundant copies are excluded from the $\mathcal{C}_{m\times n}$.

Note that, compared with Algorithm 1, Algorithm 2 only needs to perform $p = \min\{m, l, n\}$ loops (line 19) instead of l. We assume that the proper order is adopted when flattening the matrix: When $p = \min\{m, l, n\} = m$, $\mathcal{A}$ is expanded and the column-major order is adopted to flatten matrices. When $p = \min\{m, l, n\} = n$, $\mathcal{B}$ is expanded and the row-major order is adopted to flatten matrices. When $p = \min\{m, l, n\} = l$, neither $\mathcal{A}$ and $\mathcal{B}$ is expanded, and either major order can be adopted to flatten matrices.

Table 3 lists more details about the time complexity of Algorithm 2 (assuming σ and τ operations are performed on ciphertext). For line 16, from Theorems 3.1, there are $2\mu = 2\min(m, l) - 1$ nonzero diagonals in a permutation matrix, and each diagonal calls for one HE-add, one HE-CMult, and one HE-Rot operations during the linear transformation (see Sect. 3.2.1). The same reasoning can be applied for line 17 based on Theorem 3.2. For $\epsilon^0_{M\times\max(l,N)}$ and $\omega^0_{\max(l,M)\times N}$ transformation, it needs $\left\lfloor \frac{\max(l,N)}{l} \right\rfloor + 1$ and $\left(\left\lfloor \frac{\max(l,M)}{l} \right\rfloor + 2\cdot \right) N$, as explained in Theorem 3.3 and 3.4. Note that σ and τ transformations in Lines 16 and 17 only need to be performed once, which can be done on the client side on the plaintext without compromising the security/privacy of some applications. In that case, the time complexity of Lines 16 and 17 can be further reduced to 0.

For Line 20 and 21, Theorems 3.3 and 3.4, the ϵ and ω transformation require $\left\lfloor \frac{n}{l} \right\rfloor + 1$ and $(\left\lfloor \frac{m}{l} \right\rfloor + 2\cdot)n$ HE-Add, HE-CMult and HE-Rot operations, respectively. There are l loops. Lines 22–26 are a loop to compute the partial products and accumulate them together, with each iteration requiring one HE-Mult and one HE-Add operation.

Similar to Algorithm 1, Algorithm 2 has a HE multiplication depth of **1 M+2C** with one HE multiplication (e.g., **1 M**) and two constant multiplication (e.g., **2C**) as illustrated in Table 3.

Table 3 Time Complexity of HEGMM-En

Line	HE-Add	HE-Mult	He-CMult	HE-Rot	Depth
16	$\sigma : 2\mu - 1$ $\epsilon^0 : \mathcal{E}$	0	$2\mu - 1$ $\epsilon^0 : \mathcal{E}$	$2\mu - 1$ $\epsilon^0 : \mathcal{E}$	$1C$
17	$\tau : 2\nu - 1$ $\omega^0 : \mathcal{W}$	0	$2\nu - 1$ $\omega^0 : \mathcal{W}$	$2\nu - 1$ $\omega^0 : \mathcal{W}$	–
20	$2p$	0	$2p$	$2p$	$1C$
21	$2p$	0	$2p$	$2p$	–
22	p	p	0	0	$1M$
Total	$5p + \mathcal{E} + \mathcal{W}$ $+2(\mu + \nu)$	p	$4p + \mathcal{E} + \mathcal{W}$ $+2(\mu + \nu)$	$4p + \mathcal{E} + \mathcal{W}$ $+2(\mu + \nu)$	$1M + 2C$

$p = \min(m, l, n)$, $\mu = \min(m, l)$, $\nu = \min(n, l)$, $\mathcal{E} : \left\lfloor \frac{\max(l,N)}{l} \right\rfloor + 1$ and $\mathcal{W} : (\left\lfloor \frac{\max(l,M)}{l} \right\rfloor + 2\cdot)N$

4 Experiments

In this section, we evaluate the performance of the two algorithms developed in this paper, i.e., HEGMM and HEGMM-Enhanced, and compare them with the state-of-the-art schemes for HE-based matrix multiplication.

4.1 Experimental platform

We implemented HEGMM and HEGMM-Enhanced using a Python HE library, named Pyfhel [10] with BFV scheme [11, 12]. We set the HE scheme based on the RLWE (Ring Learning With Errors) [31] assumption over the cyclotomic ring $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ with $N = 2^{12}$. Thus, each ciphertext can hold up to $N = 2^{12}$ slots for plaintext values; the largest square matrix that can be accommodated in one ciphertext is thus 64×64.

In our experiments, we studied the following approaches.

- E2DM-S, which is presented in [27] on square matrix multiplication. For a general MM $\mathcal{A}_{m\times l} \times \mathcal{B}_{l\times n}$, we can transform $\mathcal{A}_{m\times l}$ and $\mathcal{B}_{l\times n}$ to two square matrices, $\mathcal{A}\simeq_{d\times d}$ and $\mathcal{B}\simeq_{d\times d}$ with $d = \max\{m, l, n\}$ and use this algorithm to calculate the result;
- E2DM-R, which is presented in [27] on rectangular matrix multiplication $\mathcal{A}_{r\times d} \times \mathcal{B}_{d\times d}$. For a general MM $\mathcal{A}_{m\times l} \times \mathcal{B}_{l\times n}$, we can expand $\mathcal{A}_{m\times l}$ and/or $\mathcal{B}_{l\times n}$ accordingly and use this algorithm to calculate the result;
- Huang et al., which is introduced in [30] and implemented with Pyfhel [10].
- uSCORE, which is introduced in [28] and implemented with Pyfhel [10].
- HEGMM, which is shown in Algorithm 1.
- HEGMM-En, which is shown in Algorithm 2.

In what follows, we first discuss and compare the computation complexities of different approaches in Sect. 4.2. We then use experimental results to compare their computational performance in Sect. 4.3 and 4.4. Finally, we compare the memory usage for different approaches in Sect. 4.5.

4.2 Time complexity analysis

Table 4 summarizes the symbolic computation complexities of different approaches. 1. As shown in the table, the computation cost for HE-Mult is the most computation costly operations, and therefore, reducing its operation numbers can help to significantly reduce the computational cost. 2. Our approach can take advantage the matrix shapes to improve performance. The further away the matrix share deviate from the square format, the better. For example, for a MM of $\mathcal{A}_{50\times 30} \times \mathcal{B}_{30\times 10}$, HEGMM-En only needs to perform 10 HE-Mult while it is 30 for E2DM and Huang et al. 3. Table 4 also shows that all different approaches with the similar multiplication

depth, with Huang et al. slightly better. In what follows, we use experimental results to further compare their performance.

4.3 Computational time evaluation

In this section, we use four different groups of experimental results to compare the computation efficiency of our approach with the state of the art. We first use experimental results, based on randomly generated metrics to evaluate the computational performance of E2DM-S, E2DM-R, Huang et al., HEGMM, and HEGMM-En, with the preprocessing work (i.e., σ and τ transformations transformations in E2DM, HEGMM, and HEGMM-En) done on the cloud (Sect. 4.3.1) or on the client side (Sect. 4.3.2). Note that since uSCORE requires dividing the source matrices into different blocks and determining the blocking method is nontrivial and may have profound impacts on its efficiencies. We therefore excluded it in our performance comparisons based on randomly generated test cases. Instead, we conducted a number of experiments on predefined matrix dimensions and compared its performance with others (Sect. 4.3.3). We further study the computing performance of different approaches for large matrices, which cannot be accommodated using one ciphertext (Sect. 4.4).

For our randomly generated test cases, we randomly generated 2000 pairs of matrices, with column and row numbers evenly distributed with [1, 64]. Note that, even though Huang et al. [30], HEGMM, and HEGMM-Enhanced can handle MM with column or row numbers exceeding 64, as long as the total element is no more than 2^{12}, we limited the largest dimension size to 64 so that E2DM-S and E2DM-R can always apply. All experiments were conducted on a server with Intel Xeon Silver 4114 at 2.2GHz.

Table 4 Comparison of time complexity among different algorithms where $d = \max(m, l, n)$, $p = \min(m, l, n)$, $s = \max(l, n)$, $\mu = \min(m, l)$, $v = \min(l, n)$, $\mathcal{E} : \left\lfloor \frac{n}{l} \right\rfloor + 1$ and $\mathcal{W} : (\left\lfloor \frac{m}{l} \right\rfloor + 2\cdot)n$

Algorithms	HE-Add	HE-Mult	He-CMult	HE-Rot	Depth
E2DM-S	$6d$	d	$4d$	$3d + 5\sqrt{d}$	1 M+2C
E2DM-R	$3d + 2p$ $+log(d/p)$	p	$3d + 2p$	$3p + 5\sqrt{d}$ $+log(d/p)$	1 M+2C
Huang et al.	$l \cdot logs + l$ $+log(d/l)$	l	l	$l \cdot logs + l$ $+log(d/l)$	1 M+1C
HEGMM	$\mathcal{E}l + \mathcal{W}l$ $+2(\mu + v) + l$	l	$\mathcal{E}l + \mathcal{W}l$ $+2(\mu + v)$	$\mathcal{E}l + \mathcal{W}l$ $+2(\mu + v)$	1 M+2C
HEGMM-En	$5p + \mathcal{E} + \mathcal{W}$ $+2(\mu + v)$	p	$4p + \mathcal{E} + \mathcal{W}$ $+2(\mu + v)$	$4p + \mathcal{E} + \mathcal{W}$ $+2(\mu + v)$	1 M+2C
Time (ms)	0.55	20.87	4.14	5.35	

4.3.1 HE MM with preprocessing on the cloud

In this group of tests, we assume that clients encrypt the source matrices and send them to the cloud. All matrix operations are performed on the cloud. To better understand the performance of different approaches, we categorize the test cases into five groups: (1) $m = \min\{m, l, n\}$; (2) $l = \min\{m, l, n\}$; (3) $n = \min\{m, l, n\}$; (4) $(m \times l) \times (l \times l)$ and $l \bmod m = 0$; (5) $m = l = n$ (the square matrix). Note that cases in (4) and (5) are the most favorable ones for E2DM-R and E2DM-S, respectively. For test cases in each group, execution times were collected for the five different approaches. We use the better ones by E2DM-S and E2DM-R as the performance that can achieved by E2DM. We then calculate the speedups that can be achieved using HEGMM, HEGMM-En over E2DM, and Huang et al. The average, median, and maximum speedup for each group, as well as the overall results, are listed in Table 5.

As in Table 5, HEGMM-En outperforms Huang et al. in all groups, with a speedup of 3.96 on average and the maximum of over 11.60. Compared with E2DM, HEGMM-En can achieve better performance in all cases other than if the matrices are square or when $(m \times l) \times (l \times l)$ and $l \bmod m = 0$. As shown in Table 5, HEGMM-En can achieve a speedup of 8.63 on average with a maximum of over 184.76 over the best of E2DM. This is because HEGMM-En can reduce HE-Mult operations significantly by properly duplicating the source matrices. When source matrices are square, HEGMM-En is equivalent to E2DM-S with slight overhead for taking care of the generality of matrices. When $(m \times l) \times (l \times l)$ and $l \bmod m = 0$, the time complexity of E2DM-R is $\mathcal{O}(m)$ while HEGMM-En is $\mathcal{O}(m)$ as well. Therefore, the speedup of them is nearly 1.00.

We also use Fig. 7 to compare the performance of these approaches from a different perspective. Specifically, Fig. 7 shows the number of test cases that can achieve speedups between (0, 1], [1, 2], and $(2, +\infty)$ by HEGMM, HEGMM-En, and Huang et al. over the best results by E2DM-S and E2DM-R. In a total of 2000 test cases, there are 1074 cases that HEGMM outperforms both E2DM-S and E2DM-R, while it is 1994 for HEGMM-En, which indicates that HEGMM-En performs significantly better than HEGMM. For Huang et al., 1964 samples outperform E2DM which is also better than HEGMM but not as good as HEGMM-En. Overall, the experimental findings indicate that the algorithm HEGMM-En exhibits a significant performance superiority compared to current methodologies in 99.7% of the samples.

4.3.2 HE MM with preprocessing on the client

In this section, we assume that preprocessing transformations (*i.e.*, σ and τ transformations) can be performed by the client on plaintext. Since this can be done as long as the dimensions of the matrices are known to the clients, the preprocessing work can be done by the clients without compromising the privacy/security of the matrices themselves. This option can further improve the computation efficiency for E2DM, HEGMM, and HEGMM-En significantly.

Table 5 The performance comparison of HEGMM-En, E2DM [27], and Huang et al. [30] in different scenarios with σ and τ are performed on ciphertext

	Speedup	Average	Median	Max
$m = min(m, l, n)$	HEGMM v.s. E2DM	1.06	0.63	14.93
	HEGMM v.s. Huang et al.	0.54	0.46	3.16
	HEGMM-En v.s. E2DM	6.70	4.56	98.45
	HEGMM-En v.s. Huang et al.	3.60	3.49	10.25
$l = min(m, l, n)$	HEGMM v.s. E2DM	1.80	1.15	23.72
	HEGMM v.s. Huang et al.	0.82	0.85	1.69
	HEGMM-En v.s. E2DM	9.51	7.89	50.06
	HEGMM-En v.s. Huang et al.	5.04	4.80	11.60
$n = min(m, l, n)$	HEGMM v.s. E2DM	2.79	1.69	66.37
	HEGMM v.s. Huang et al.	0.93	0.84	4.21
	HEGMM-En v.s. E2DM	9.82	7.16	184.76
	HEGMM-En v.s. Huang et al.	3.36	3.07	9.78
$(m \times l) \times (l \times l)$	HEGMM v.s. E2DM	0.74	0.46	1.00
	HEGMM v.s. Huang et al.	0.84	1.47	2.06
$l \mod m = 0$	HEGMM-En v.s. E2DM	0.99	0.99	1.08
	HEGMM-En v.s. Huang et al.	1.17	3.31	6.19
square	HEGMM v.s. E2DM	1.00	1.00	1.02
	HEGMM v.s. Huang et al.	0.92	0.95	1.21
	HEGMM-En v.s. E2DM	0.98	0.99	1.00
	HEGMM-En v.s. Huang et al.	0.90	0.95	1.21
overall	HEGMM v.s. E2DM	1.88	1.07	66.37
	HEGMM v.s. Huang et al.	0.76	0.70	4.21
	HEGMM-En v.s. E2DM	8.63	6.10	184.76
	HEGMM-En v.s. Huang et al.	3.96	3.54	11.60

The test cases for this group of experiments were generated as before, and the average, median, and maximum speedup for each group, as well as the overall results, are listed in Table 6.

As in Table 6, HEGMM outperforms Huang *el al.* in all groups, with a speedup of 1.93 on average and the maximum of over 4.96. Compared with E2DM, HEGMM can achieve better performance in all cases other than if the matrices are square or when $m = min(m, l, n)$. As shown in Table 6, HEGMM can achieve a speedup of 3.3 on average with a maximum of over 154.12 over the best of E2DM. When source matrices are square, HEGMM is equivalent to E2DM with slight overhead for taking care of the generality of matrices. When $m = min(m, l, n)$, the time complexity of E2DM-R is $\mathcal{O}(m)$ while HEGMM is $\mathcal{O}(l)$. Therefore, E2DM-R can potentially achieve better performance, especially when $m << l$.

The enhanced algorithm, i.e., HEGMM-En, can still significantly outperform the rest of the approaches for arbitrary HE MM, as shown in Table 6. This is

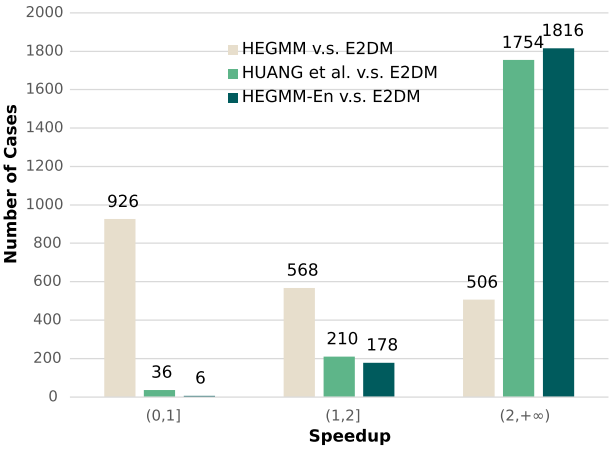


Fig. 6 The statistics of the speedups for the algorithms HEGMM, HEGMM-En, E2DM [27], and Huang et al. [30] when σ and τ are performed on ciphertext

Table 6 The performance comparison of HEGMM-En, E2DM [27], and Huang et al. [30] in different scenarios with σ and τ are performed on plaintext

	Speedup	Average	Median	Max
$m = min(m, l, n)$	HEGMM v.s. E2DM	0.90	0.58	39.50
	HEGMM v.s. Huang et al.	2.21	2.21	3.25
	HEGMM-En v.s. E2DM	1.69	1.13	33.28
	HEGMM-En v.s. Huang et al.	6.60	4.88	23.29
$l = min(m, l, n)$	HEGMM v.s. E2DM	10.74	3.61	154.12
	HEGMM v.s. Huang et al.	1.99	1.95	4.96
	HEGMM-En v.s. E2DM	10.32	3.70	132.42
	HEGMM-En v.s. Huang et al.	2.01	1.97	4.26
$n = min(m, l, n)$	HEGMM v.s. E2DM	1.83	1.20	136.82
	HEGMM v.s. Huang et al.	2.29	2.35	3.28
	HEGMM-En v.s. E2DM	4.06	2.56	113.31
	HEGMM-En v.s. Huang et al.	6.55	4.83	23.68
$(m \times l) \times (l \times l)$	HEGMM v.s. E2DM	1.51	1.47	2.06
	HEGMM v.s. Huang et al.	0.53	0.46	1.00
$l \mod m = 0$	HEGMM-En v.s. E2DM	0.99	0.99	1.08
	HEGMM-En v.s. Huang et al.	3.45	6.23	21.75
square	HEGMM v.s. E2DM	1.00	1.00	1.02
	HEGMM v.s. Huang et al.	1.67	1.72	2.36
	HEGMM-En v.s. E2DM	0.99	1.00	1.01
	HEGMM-En v.s. Huang et al.	1.66	1.72	2.36
overall	HEGMM v.s. E2DM	3.30	1.04	154.12
	HEGMM v.s. Huang et al.	1.93	2.04	4.96
	HEGMM-En v.s. E2DM	4.13	1.38	132.42
	HEGMM-En v.s. Huang et al.	4.50	2.48	23.68

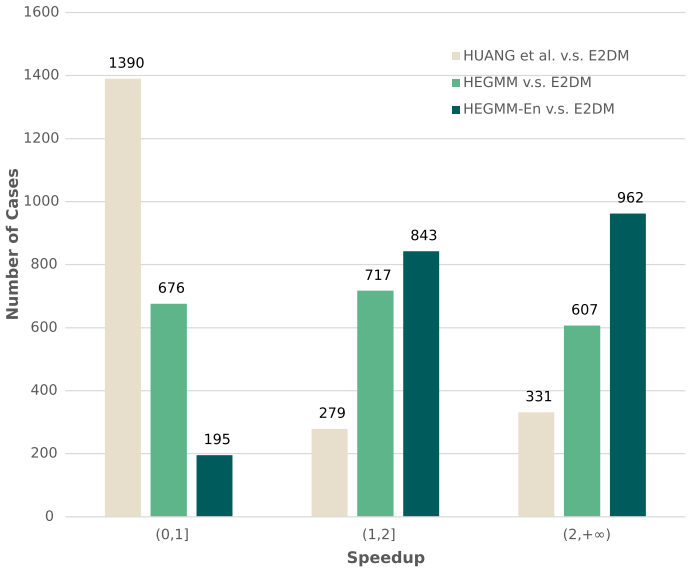


Fig. 7 The statistics of the speedups for the algorithms HEGMM, HEGMM-En, E2DM [27], and Huang et al. [30] when σ and τ are performed on plaintext

because HEGMM-En can reduce HE-Mult operations significantly by properly duplicating the source matrices. Specifically, HEGMM-En can achieve an average speedup of 4.13 with a maximum of 132.42 over the best of E2DM and an average speedup of 4.50 with a maximum of 23.68 over the Huang et al. For square matrices, HEGMM-En is equivalent to E2DM and requires slightly more time than due to the overhead for taking care of the generality of matrices.

We also use Fig. 6 to compare the performance of these approaches from a different perspective. Specifically, Fig. 6 shows the number of test cases that can achieve speedups between (0, 1], [1, 2], and $(2, +\infty)$ by HEGMM, HEGMM-En, and Huang et al. over the best results by E2DM-S and E2DM-R. In a total of 2000 test cases, there were 1324 cases in which HEGMM outperformed both E2DM-S and E2DM-R, while it is 1805 for HEGMM-En, which indicates that HEGMM-En performs significantly better than HEGMM. For Huang et al., only 610 samples outperform E2DM. More results of HE MM with some specific dimensions are shown in Table 10 in Appendix 3.

Overall, the experimental results clearly show that HEGMM and HEGMM-En exhibit a significant performance superiority compared to current methodologies in 66.2% and 90.2% of the samples, respectively.

4.3.3 HE MM with predefined dimensions

As mentioned before, uSCORE [28] relies on matrix blocking for MM, and designing the appropriate blocking methods for a given matrix dimension is nontrivial and has great potential to affect its computing performance. We, therefore, chose a set

of matrices that can fit in one ciphertext, with predefined dimensions (as shown in Table 7) and block size to compare its performance with others. For each test case, we repeated the experiments 10 times and the average results were collected and shown in Table 7.

As shown in Table 7, HEGMM-En achieves or comes very close to the best performance. This is because HEGMM-En's time complexity (dominate part which is HE-Mult) is $\mathcal{O}(16)$ while E2DM-S is $\mathcal{O}(64)$ or $\mathcal{O}(128)$. For E2DM-R, its performance boosts only when matrix size is 16–64–64 and 16–128–128. This is because its time complexity is $\mathcal{O}(16)$ which is consistent with our analysis in Sect. 4.2. HEGMM and Huang et al. are similar because their time complexity is $\mathcal{O}(l)$ (l is the middle number in matrix size).

For uSCORE [28], which is similar to E2DM but includes a blocking algorithm, the processing time increases when blocking is unnecessary. In all six groups of matrices, each matrix fits into a single ciphertext, which negates the advantages of uSCORE. When matrices are small enough to fit into one ciphertext, uSCORE requires more operations (as shown in Fig. 4 of uSCORE), resulting in longer processing times compared to both E2DM and our proposed algorithm. Conversely, for matrices too large to fit into one ciphertext, uSCORE outperforms E2DM. This is because E2DM's blocking method, which involves dividing the matrix into smaller square matrices, becomes inefficient. On the other hand, for HEGMM, there are numerous ways to block the matrix, presenting a separate research challenge.

In summary, when the matrix size is small enough to fit into a single ciphertext, our approach significantly outperforms uSCORE [28]. However, for larger matrices where blocking is necessary, uSCORE [28] proves to be very efficient. Nonetheless, designing optimal blocking methods for the source matrices remains a critical challenge.

4.4 HE MM for large matrices

Our test cases above are limited to the maximum matrix dimension of 64x64, the largest one that can fit into one ciphertext in our setting. When matrix sizes

Table 7 Performance comparison

time(s)	64–64–16	16–64–64	64–16–64	128–128–16	16–128–128	128–16–128
E2DM-S	6.07	6.12	6.08	273.57	275.00	277.20
E2DM-R	6.06	**3.10**	N/A	273.73	52.18	N/A
uSCORE	5.94	5.96	5.93	363.16	364.98	370.66
Huang et al.	5.81	5.87	3.56	432.14	433.21	49.75
HEGMM	5.88	6.08	**3.07**	267.43	274.41	49.62
HEGMM-En	**3.12**	3.13	3.08	**54.14**	**52.09**	**49.58**

Size 64 is with $N = 2^{13}$ which means ciphertext has 4096 slots; size 128 is with $N = 2^{15}$ which means ciphertext has 16,384 slots

Bolded numbers represent the best performance

exceed this limit, we can resort to the traditional blocking algorithm, i.e., by dividing a large matrix into a series of smaller blocks, to perform the MM calculation. We want to study the performance of our proposed approaches when incorporated into MM blocking algorithms for $\mathcal{A}_{100\times100} \times \mathcal{B}_{100\times100}$.

Partitioning large source matrices properly based on different MM algorithms is an interesting problem but beyond the scope of this paper. In our experiments, we hire two intuitive partition methods: P1: partitioning the matrix 100×100 to four equal-size square matrices of 50×50; P2: partitioning the matrix 100×100 to four sub-matrices of 64×64, 64×36, 36×64, and 36×36.

Different HE MM algorithms were employed for blocking MMs. We ran the experiments 10 times, and the average results were collected and shown in Table 8. As expected, for P1 when all matrices are square, E2DM-S, E2DM-R, HEGMM, and HEGMM-En perform quite similarly, while HEGMM and HEGMM-En take a little longer due to overhead in dealing with the generality of the matrices. Huang et al. showed a much slower performance than the others. We believe this is because that Huang et al. approach requires duplicating diagonals of a source matrix with the complexity of $O(logN)$, with N the size of the matrix. The duplication operation involves expensive HE-CMult and HE-Rot operations. This is particularly computationally expensive when N is not a power of two. In contrast, the time complexity of same step in E2DM and HEGMM is $O(2)$ for P1.

For $P2$, HEGMM, HEGMM-En, and Huang et al. can perform better because they can take advantage of the irregular shapes of the matrices. In particular, HEGMM-En (resp. HEGMM) has a complexity of $O(\min(m, l, n))$ (resp. $O(l)$). In contrast, E2DM-S runs much longer because it needs to expand matrices 64×36 and 36×64 to form 64×64 matrix. E2DM-R is incapable of processing matrices with such irregular shapes, as it has a tendency to enlarge matrix of 36×64 to 72×72, which is larger than the ciphertext size.

4.5 Memory evaluations

HE computations may demand not only excessive computation time but also memory usage as well. We are therefore interested in studying the memory usage of these approaches. We collected the memory usage for each algorithm during its runtime for our test cases with results normalized against the memory usage by E2DM and presented in Fig. 8, where a total of 2000 experimental sets were conducted. In comparison with E2DM, both HEGMM and HEGMM-En tend to consume less memory. As shown in Fig. 8, less than 17 (resp. 30) out of the total

Table 8 Time evaluation of the blocking algorithm

Partition	E2DM-S	E2DM-R	Huang et al.	HEGMM	HEGMM-En
P1	39.06s	**39.01s**	74.34s	39.12s	39.15s
P2	29.76s	N/A	37.51s	**26.17s**	26.23s

Bolded numbers represent the best performance

2000 test results show that E2DM consumes less memory than HEGMM (resp. HEGMM-En). In contrast, 630 test cases using Huang et al. have higher memory usage compared to E2DM. Overall, the experimental results clearly demonstrate the advantage of memory usage efficiency of HEGMM and HEGMM-En over the existing approaches.

5 Conclusions

HE has great potential for security and privacy protection when outsourcing data processing to the cloud. However, the excessive computational overhead associated with the HE operations makes it prohibitive for many practical cloud applications. We study how to reduce the HE computational cost for general MM operation, an essential building block in many computational fields. We present two HE MM algorithms, with one improving another, to reduce the computational complexity of MM by taking advantage of the SIMD structure in the HE scheme. We also conduct rigorous analytical studies on the correctness and computational complexity of these two algorithms. Experiment results show that our proposed approach can significantly outperform the existing methods. We want to mention that our approach show excellent performance for matrix size that can fit into one ciphertext. However, how to do blocking is another interesting problem to study. In addition, to incorporate our approach with the blocking method in USCORE seems an interesting problem to pursue. Moreover, in our future research, we plan to investigate how to reduce the HE computational cost for sparse matrix multiplication.

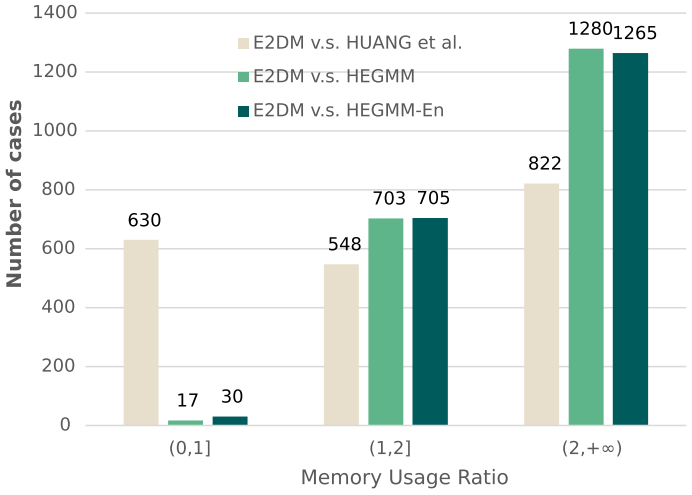


Fig. 8 The statistics of memory usage ratio for the algorithms HEGMM, HEGMM-En, E2DM [27], and Huang et al. [30]

Appendix 1: The proof for Theorem 3.1 to Theorem 3.4

Theorem 3.1 *Let $\sigma(\mathcal{A}) = \mathbf{U}^{\sigma}\mathcal{A}$ for $\mathcal{A}$ with a dimension of $m \times l$. There are at most $2 \cdot \min(m, l) - 1$ nonzero diagonals in $\mathbf{U}^{\sigma}$ no matter if the matrix is flattened with a column-major or row-major order.*

Proof When applying σ transformation on matrix $\mathcal{A}_{m\times l}$ in column-major order, $\mathbf{U}^{\sigma}$ is formulated in Eq. (12). Note that $\mathbf{U}^{\sigma}_{i+j\cdot m,h} = 1$ when $h = i + [i+j]_l \cdot m$ and, for all elements of $\mathbf{U}^{\sigma}_{i+j\cdot m,h}$ that belong to the same diagonal, we have $h - (i + j \cdot m)$ as a constant.

Considering all the nonzero elements in $\mathbf{U}^{\sigma}_{i+j\cdot m,h}$, we have

$$\begin{aligned} h - (i + j \cdot m) &= i + [i+j]_l \cdot m - (i + j \cdot m) \\ &= i + (i + j - \left\lfloor \frac{i+j}{l} \right\rfloor \cdot l) \cdot m - (i + j \cdot m) \\ &= (i - \left\lfloor \frac{i+j}{l} \right\rfloor \cdot l) \cdot m. \end{aligned}$$

Since $\left\lfloor \frac{i}{l} \right\rfloor + \left\lfloor \frac{j}{l} \right\rfloor \le \left\lfloor \frac{i+j}{l} \right\rfloor \le \left\lfloor \frac{i}{l} \right\rfloor + \left\lfloor \frac{j}{l} \right\rfloor + 1$ and $0 \le j < l$, we have $\left\lfloor \frac{i}{l} \right\rfloor \le \left\lfloor \frac{i+j}{l} \right\rfloor \le \left\lfloor \frac{i}{l} \right\rfloor + 1$.

Now consider two different scenarios: 1) $m < l$; 2) $m \ge l$. When $m < l$, for each $i = \{1, 2, \ldots, m-1\}$, $h - (i + j \cdot m)$ can at most take two constant values since $\left\lfloor \frac{i}{l} \right\rfloor = 0$ and $0 \le \left\lfloor \frac{i+j}{l} \right\rfloor \le 1$. When $i = 0$, $h - (i + j \cdot m)$ can only be zero since $\left\lfloor \frac{i+j}{l} \right\rfloor = 0$. Therefore, $\mathbf{U}^{\sigma}_{i+j\cdot m,h}$ has at most $2m - 1$ nonzero diagonals under this case.

When $m \ge l$, we have

$$\begin{aligned} h - (i + j \cdot m) &= \left(i - \left\lfloor \frac{i+j}{l} \right\rfloor \cdot l \right) \cdot m \\ &= \left(\left\lfloor \frac{i}{l} \right\rfloor \cdot l + p - \left\lfloor \frac{i+j}{l} \right\rfloor \cdot l \right) \cdot m, \end{aligned}$$

with $0 \le p < l$. Since $-1 \le \left(\left\lfloor \frac{i}{l} \right\rfloor - \left\lfloor \frac{i+j}{l} \right\rfloor \right) \le 0$, $\mathbf{U}^{\sigma}_{i+j\cdot m,h}$ has at most $2l - 1$ nonzero diagonals under this case.

Therefore, in summary, there are at most $2 \cdot \min(m, l) - 1$ nonzero diagonals in $\mathbf{U}^{\sigma}$ when the matrix is flattened with a column-major. Similar proof can be obtained when the matrix is flattened with the row-major order. □

Theorem 3.2 *Let $\tau(\mathcal{B}) = \mathbf{U}^{\tau}\mathcal{B}$ for $\mathcal{B}$ with a dimension of $l \times n$. There are at most $2 \cdot \min(n, l) - 1$ nonzero diagonals in $\mathbf{U}^{\tau}$ no matter if the matrix is flattened with a column-major or row-major order.*

Proof When applying τ transformation on matrix $\mathcal{B}_{l\times n}$ in column-major order, $\mathbf{U}^{\tau}$ is formulated in Eq. (13). Note that $\mathbf{U}^{\tau}_{i+j\cdot l,h} = 1$ when $h = [i+j]_l + j\cdot l$ and, for all elements of $\mathbf{U}^{\tau}_{i+j\cdot l,h}$ that belong to the same diagonal, we have $h-(i+j\cdot l)$ as a constant.

Considering all the nonzero elements in $\mathbf{U}^{\tau}_{i+j\cdot l,h}$, we have

$$\begin{aligned} h-(i+j\cdot l) &= [i+j]_l + j\cdot l - (i+j\cdot m) \\ &= i+j-\left\lfloor \frac{i+j}{l} \right\rfloor \cdot l + j\cdot l - (i+j\cdot m) \\ &= j - \left\lfloor \frac{i+j}{l} \right\rfloor \cdot l. \end{aligned}$$

Since $\left\lfloor \frac{i}{l} \right\rfloor + \left\lfloor \frac{j}{l} \right\rfloor \le \left\lfloor \frac{i+j}{l} \right\rfloor \le \left\lfloor \frac{i}{l} \right\rfloor + \left\lfloor \frac{j}{l} \right\rfloor + 1$ and $0 \le i < l$, we have $\left\lfloor \frac{j}{l} \right\rfloor \le \left\lfloor \frac{i+j}{l} \right\rfloor \le \left\lfloor \frac{j}{l} \right\rfloor + 1$.

Now consider two different scenarios: 1) $n < l$; 2) $n \ge l$. When $n < l$, for each $j = \{1, 2, \ldots, n-1\}$, $h-(i+j\cdot l)$ can at most take two constant values since $\left\lfloor \frac{j}{l} \right\rfloor = 0$ and $0 \le \left\lfloor \frac{i+j}{l} \right\rfloor \le 1$. When $i = 0$, $h-(i+j\cdot l)$ can only be zero since $\left\lfloor \frac{i+j}{l} \right\rfloor = 0$. Therefore, $\mathbf{U}^{\tau}_{i+j\cdot l,h}$ has at most $2n-1$ nonzero diagonals under this case.

When $n \ge l$, we have

$$\begin{aligned} h-(i+j\cdot l) &= j - \left\lfloor \frac{i+j}{l} \right\rfloor \cdot l \\ &= \left\lfloor \frac{j}{l} \right\rfloor \cdot l + p - \left\lfloor \frac{i+j}{l} \right\rfloor \cdot l, \end{aligned}$$

with $0 \le p < l$. Since $-1 \le (\left\lfloor \frac{j}{l} \right\rfloor - \left\lfloor \frac{i+j}{l} \right\rfloor) \le 0$, $\mathbf{U}^{\tau}_{i+j\cdot l,h}$ has at most $2l-1$ nonzero diagonals under this case.

Therefore, in summary, there are at most $2\cdot\min(n,l)-1$ nonzero diagonals in $\mathbf{U}^{\tau}$ when the matrix is flattened with a column-major. Similar proof can be obtained when the matrix is flattened with the row-major order. □

Theorem 3.3 *Let $\epsilon^k_{m\times n}(\mathcal{A}) = \mathbf{U}^{\epsilon^k_{m\times n}}\mathcal{A}$ be the linear transformation $\epsilon_{m\times n} : \mathcal{R}_{m\times l} \to \mathcal{R}_{m\times n}$ with matrix $\mathcal{A}$ having a dimension of $m\times l$. There are at most $\left\lfloor \frac{n}{l} \right\rfloor + 1$ nonzero diagonal vectors in $\mathbf{U}^{\epsilon^k_{m\times n}}$ when the matrix is flattened with the column-major order. There are at most $(\left\lfloor \frac{n}{l} \right\rfloor + 2)\cdot m$ nonzero diagonal vectors in $\mathbf{U}^{\epsilon^k_{m\times n}}$ when matrix $\mathcal{A}$ is flattened with the row-major order. Specifically, when $n = l$, there are no more than two nonzero diagonals in $\mathbf{U}^{\epsilon^k_{m\times n}}$, no matter if the matrix is flattened in column-major or row-major order.*

Proof When applying ϵ transformation on matrix $\mathcal{A}_{m\times l}$ in column-major order, $\mathbf{U}^{\epsilon}$ is formulated in Eq. (14). Note that $\mathbf{U}^{\epsilon^k_{m\times n}}_{i,j} = 1$ when $j = [k\cdot m + i]_{m\cdot l}$ and, for all elements of $\mathbf{U}^{\epsilon^k_{m\times n}}_{i,j}$ that belong to the same diagonal, we have $j-i$ as a constant.

Considering all the nonzero elements in $\mathbf{U}^{\epsilon^k_{m\times n}}_{i,j}$, we have

$$\begin{aligned} j-i &= [k\cdot m+i]_{m\cdot l}-i \\ &= k\cdot m+i-\left\lfloor\frac{k\cdot m+i}{m\cdot l}\right\rfloor\cdot m\cdot l-i \\ &= k\cdot m-\left\lfloor\frac{k\cdot m+i}{m\cdot l}\right\rfloor\cdot m\cdot l \end{aligned}$$

Since $\max(k)=l-1$ and $\max(i)=m\cdot n-1$, we have

$$\begin{aligned} \max(\frac{k\cdot m+i}{m\cdot l}) &< \frac{l-1+n}{l} \\ &\leq \left\lfloor\frac{l-1}{l}\right\rfloor+\left\lfloor\frac{n}{l}\right\rfloor+1 \\ &= \left\lfloor\frac{n}{l}\right\rfloor+1 \end{aligned}$$

Therefore, we get $\left\lfloor\frac{k\cdot m+i}{m\cdot l}\right\rfloor\in\{0,1,\ldots,\left\lfloor\frac{n}{l}\right\rfloor\}$. Then, $j-i=k\cdot m-\left\lfloor\frac{k\cdot m+i}{m\cdot l}\right\rfloor\cdot m\cdot l$. k, m and l are all constant number for one transformation. The set $\{0,1,\ldots,\left\lfloor\frac{n}{l}\right\rfloor\}$ is of size $\left\lfloor\frac{n}{l}\right\rfloor+1$. In summary, $\mathbf{U}^{\epsilon^k_{m\times n}}$ has at most $\left\lfloor\frac{n}{l}\right\rfloor+1$ constant values when $\mathcal{A}_{m\times l}$ in column-major.

Special circumstances are when $n=l$, $\left\lfloor\frac{n}{l}\right\rfloor=1$. Therefore, $\left\lfloor\frac{n}{l}\right\rfloor+1=2$ and this means $\mathbf{U}^{\epsilon^k_{m\times n}}$ has only **2** nonzero diagonals when $n=l$..

When applying ϵ transformation on matrix $\mathcal{A}_{m\times l}$ in row-major order, we can formulate permutation matrix according to formula (15), but apply on $\mathcal{A}_{l\times m}$ instead of $\mathcal{A}_{l\times n}$. Note that $\mathbf{U}^{\epsilon^k_{m\times n}}_{i,j}=1$ when $j=[k+[i]_n]_l+\lfloor i/n\rfloor\cdot l$ and, for all elements of $\mathbf{U}^{\epsilon^k_{m\times n}}_{i,j}$ that belong to the same diagonal, we have $j-i$ as a constant.

Considering all the nonzero elements in $\mathbf{U}^{\epsilon^k_{m\times n}}_{i,j}$, we have

$$\begin{aligned} j &= k+[i]_n-\left\lfloor\frac{k+[i]_n}{l}\right\rfloor\cdot l+\left\lfloor\frac{i}{n}\right\rfloor\cdot l \\ &= k+[i]_n+\left(\left\lfloor\frac{i}{n}\right\rfloor-\left\lfloor\frac{k+[i]_n}{l}\right\rfloor\right)\cdot l \end{aligned}$$

Since $i\in[0,mn)$, we split i to m circumstances that $i\in[pn,(p+1)n)$ where $p=\{0,1,2,\ldots,m-1\}$. For each circumstance that $i\in[pn,(p+1)n)$, we have

$$j=k+i-pn+\left(p-\left\lfloor\frac{k+[i]_n}{l}\right\rfloor\right)\cdot l$$

and

$$j - i = k - pn + \left(p - \left\lfloor \frac{k + [i]_n}{l} \right\rfloor\right) \cdot l$$

Note that we have

$$\left\lfloor \frac{[pn]_n}{l} \right\rfloor \leq \left\lfloor \frac{k + [i]_n}{l} \right\rfloor < \left\lfloor \frac{[pn]_n}{l} \right\rfloor + \left\lfloor \frac{n}{l} \right\rfloor + 1 + 1$$

which has $2 + \left\lfloor \frac{n}{l} \right\rfloor$ constant values. And this means $j - i$, which represents the number of nonzero diagonals in $\mathbf{U}^{\epsilon^k_{m\times n}}$, has $(2 + \left\lfloor \frac{n}{l} \right\rfloor) \cdot m$ in total when $\mathcal{A}_{m\times l}$ in row-major because there are m circumstances.

Special circumstances are when $n = l$, $j - i \in \{0, 1\}$. The reason is that, since

$$\left\lfloor \frac{k}{l} \right\rfloor + \left\lfloor \frac{[i]_n}{l} \right\rfloor \leq \left\lfloor \frac{k + [i]_n}{l} \right\rfloor \leq \left\lfloor \frac{k}{l} \right\rfloor + \left\lfloor \frac{[i]_n}{l} \right\rfloor + 1$$

and we also have $k < l$ and $[i]_n < l$, thus

$$0 \leq \left\lfloor \frac{k + [i]_n}{l} \right\rfloor \leq 1$$

On the other hand, we have

$$j - i = k - \left\lfloor \frac{k + [i]_n}{l} \right\rfloor \cdot l$$

for each $i \in [pn, (p + 1)n)$. $j - i$ has the same constant value in each $i \in [pn, (p + 1)n)$ and this means $\mathbf{U}^{\epsilon^k_{m\times n}}$ has only **2** nonzero diagonals when $n = l$. □

Theorem 3.4 *Let* $\omega^k_{m\times n}(\mathcal{B}) = \mathbf{U}^{\omega^k_{m\times n}}\mathcal{B}$ *be the linear transformation* $\omega_{m\times n} : \mathcal{R}_{l\times n} \to \mathcal{R}_{m\times n}$ *with matrix* $\mathcal{B}$ *having a dimension of* $l \times n$. *There are at most* $(\left\lfloor \frac{m}{l} \right\rfloor + 2) \cdot n$ *nonzero diagonal vectors in* $\mathbf{U}^{\omega^k_{m\times n}}$ *when the matrix is flattened with column-major order. There are at most* $\left\lfloor \frac{m}{l} \right\rfloor + 1$ *nonzero diagonal vectors in* $\mathbf{U}^{\omega^k_{m\times n}}$ *when matrix* $\mathcal{B}$ *is flattened with row-major order. Specifically, when* $m = l$, *there are no more than two nonzero diagonals in* $\mathbf{U}^{\omega^k_{m\times n}}$, *no matter if the matrix is flattened in column-major or row-major order.*

Proof When applying ω transformation on matrix $\mathcal{B}_{l\times n}$ in column-major order, $\mathbf{U}^{\omega}$ is formulated in Eq. (15). Note that $\mathbf{U}^{\omega^k_{m\times n}}_{i,j} = 1$ when $j = [k + [i]_m]_l + \lfloor i/m \rfloor \cdot l$ and, for all elements of $\mathbf{U}^{\omega^k_{m\times n}}_{i,j}$ that belong to the same diagonal, we have $j - i$ as a constant.

Considering all the nonzero elements in $\mathbf{U}^{\omega^k_{m\times n}}_{i,j}$, we have

$$
\begin{aligned}
j &= k + [i]_m - \left\lfloor \frac{k + [i]_m}{l} \right\rfloor \cdot l + \left\lfloor \frac{i}{m} \right\rfloor \cdot l \\
&= k + [i]_m + \left(\left\lfloor \frac{i}{m} \right\rfloor - \left\lfloor \frac{k + [i]_m}{l} \right\rfloor \right) \cdot l
\end{aligned}
$$

Since $i \in [0, mn)$, we split i to n circumstances that $i \in [pm, (p+1)m)$ where $p = \{0, 1, 2, \ldots, n-1\}$. For each circumstance that $i \in [pm, (p+1)m)$, we have

$$
j = k + i - pm + \left(p - \left\lfloor \frac{k + [i]_m}{l} \right\rfloor \right) \cdot l
$$

and

$$
j - i = k - pm + \left(p - \left\lfloor \frac{k + [i]_m}{l} \right\rfloor \right) \cdot l
$$

Note that we have

$$
\left\lfloor \frac{[pm]_m}{l} \right\rfloor \leq \left\lfloor \frac{k + [i]_m}{l} \right\rfloor < \left\lfloor \frac{[pm]_m}{l} \right\rfloor + \left\lfloor \frac{m}{l} \right\rfloor + 1 + 1
$$

which has $2 + \left\lfloor \frac{m}{l} \right\rfloor$ constant values. And this means $j - i$, which represents the number of nonzero diagonals in $\mathbf{U}^{\omega^k_{m\times n}}$, has $(2 + \left\lfloor \frac{m}{l} \right\rfloor) \cdot n$ in total when $\mathcal{B}_{m\times l}$ in row-major because there are n circumstances.

Special circumstances are when $m = l$, $j - i \in \{0, 1\}$. The reason is that, since

$$
\left\lfloor \frac{k}{l} \right\rfloor + \left\lfloor \frac{[i]_l}{l} \right\rfloor \leq \left\lfloor \frac{k + [i]_l}{l} \right\rfloor \leq \left\lfloor \frac{k}{l} \right\rfloor + \left\lfloor \frac{[i]_l}{l} \right\rfloor + 1
$$

and we also have $k < l$ and $[i]_l < l$, thus

$$
0 \leq \left\lfloor \frac{k + [i]_l}{l} \right\rfloor \leq 1
$$

On the other hand, we have

$$
j - i = k - \left\lfloor \frac{k + [i]_l}{l} \right\rfloor \cdot l
$$

for each $i \in [pm, (p+1)m)$. $j - i$ has the same constant value in each $i \in [pm, (p+1)m)$ and this means $\mathbf{U}^{\omega^k_{m\times n}}$ has only **2** nonzero diagonals when $m = l$.

When applying ω transformation on matrix $\mathcal{B}_{l\times n}$ in row-major order, we can formulate permutation matrix according to formula (14), but apply on $\mathcal{B}_{n\times l}$ instead of $\mathcal{B}_{m\times l}$. Note that $\mathbf{U}^{\omega^k_{m\times n}}_{i,j} = 1$ when $j = [k \cdot n + i]_{n\cdot l}$ and, for all elements of $\mathbf{U}^{\omega^k_{m\times n}}_{i,j}$ that belong to the same diagonal, we have $j - i$ as a constant.

Considering all the nonzero elements in $\mathbf{U}^{\omega^k_{m\times n}}_{i,j}$, we have

$$\begin{aligned} j - i &= [k \cdot n + i]_{n \cdot l} - i \\ &= k \cdot n + i - \left\lfloor \frac{k \cdot n + i}{n \cdot l} \right\rfloor \cdot n \cdot l - i \\ &= k \cdot n - \left\lfloor \frac{k \cdot n + i}{n \cdot l} \right\rfloor \cdot n \cdot l \end{aligned}$$

Since $\max(k) = l - 1$ and $\max(i) = m \cdot n - 1$, we have

$$\begin{aligned} \max(\frac{k \cdot n + i}{n \cdot l}) &< \frac{l - 1 + m}{l} \\ &\leq \left\lfloor \frac{l-1}{l} \right\rfloor + \left\lfloor \frac{m}{l} \right\rfloor + 1 \\ &= \left\lfloor \frac{m}{l} \right\rfloor + 1 \end{aligned}$$

Therefore, we get $\left\lfloor \frac{k \cdot n + i}{n \cdot l} \right\rfloor \in \{0, 1, \ldots, \left\lfloor \frac{m}{l} \right\rfloor\}$. Then, $j - i = k \cdot n - \left\lfloor \frac{k \cdot n + i}{n \cdot l} \right\rfloor \cdot n \cdot l$. Here, k, n and l are all constant number for one transformation. The set $\{0, 1, \ldots, \left\lfloor \frac{m}{l} \right\rfloor\}$ is of size $\left\lfloor \frac{m}{l} \right\rfloor + 1$. In summary, $\mathbf{U}^{\omega^k_{m \times n}}$ has at most $\left\lfloor \frac{m}{l} \right\rfloor + 1$ constant values when $\mathcal{B}_{m \times l}$ in row-major.

Special circumstances are when $m = l$, $\left\lfloor \frac{m}{l} \right\rfloor = 1$. Therefore, $\left\lfloor \frac{m}{l} \right\rfloor + 1 = 2$ and this means $\mathbf{U}^{\omega^k_{m \times n}}$ has only **2** nonzero diagonals when $m = l$. □

Theorem 3.5 *Let $\mathcal{A}_{m \times l}$ and $\mathcal{B}_{l \times n}$ with $m < l$, and let $\bar{A}$ be matrix expanded with $t = \left\lceil \frac{l}{m} \right\rceil$ copies of $\mathcal{A}$ vertically, i.e., $\bar{\mathcal{A}} = \{\bar{A}_0; \bar{A}_1; \ldots; \bar{A}_{(t-1)}\}^T$ with $\bar{A}_0 = \bar{A}_1 = \cdots = \bar{A}_{(t-1)} = \mathcal{A}_{m \times l}$. Then,*

- $\epsilon^k_{tm \times n}(\sigma(\bar{\mathcal{A}})) \odot \omega^k_{tm \times n}(\tau(\mathcal{B}))$ contains t items of $\epsilon^p_{m \times n}(\sigma(\mathcal{A})) \odot \omega^p_{m \times n}(\tau(\mathcal{B}))$, with $p \in \{[k]_l, [k + m]_l, \ldots, [k + (t - 1)m]_l\}$.
- $\epsilon^k_{tm \times n}(\sigma(\bar{\mathcal{A}})) \odot \omega^k_{tm \times n}(\tau(\mathcal{B}))$, $k = 0, 1, \ldots, (m - 1)$ contains all items of $\epsilon^p_{m \times n}(\sigma(\mathcal{A})) \odot \omega^p_{m \times n}(\tau(\mathcal{B}))$, with $p \in \{0, 1, \ldots, (l - 1)\}$.

Proof Consider a sub-matrix of $(\epsilon^k_{tm \times n} \circ \sigma(\bar{\mathcal{A}}))$ with dimension of $m \times n$, i.e., $(\epsilon^k_{tm \times n} \circ \sigma(\bar{\mathcal{A}}))_{hm+i,j}$, where $0 \leq i < m, 0 \leq j < n$. h is a constant with $0 \leq h < t$. Based on Eq. (1) and (3), we have

$$\begin{aligned} (\epsilon^k_{tm \times n} \circ \sigma(\bar{\mathcal{A}}))_{hm+i,j} &= \sigma(\bar{\mathcal{A}})_{hm+i,[j+k]_l} \\ &= \bar{\mathcal{A}}_{hm+i,[hm+i+j+k]_l} \\ &= \mathcal{A}_{i,[hm+i+j+k]_l} \end{aligned} \tag{16}$$

On the other hand, let $p = [k + hm]_l$, for $0 \leq i < m, 0 \leq j < n$, we have

$$\begin{aligned}(\epsilon^{p}_{m\times n}\circ\sigma(\mathcal{A}))_{i,j} &= \sigma(\mathcal{A})_{i,[j+p]_l}\\ &= \mathcal{A}_{i,[i+j+k+hm]_l}\end{aligned} \tag{17}$$

Similarly, consider the sub-matrix of $(\omega^{k}_{tm\times n}\circ\tau(\mathcal{B}))$ with dimension of $m\times n$, i.e., $(\omega^{k}_{tm\times n}\circ\tau(\mathcal{B}))_{hm+i,j}$, with $0\le i<m, 0\le j<n$. Based on Eq. (2) and (4), we have

$$\begin{aligned}(\omega^{k}_{tm\times n}\circ\tau(\mathcal{B}))_{hm+i,j} &= \tau(\mathcal{B})_{[hm+i+k]_l,j}\\ &= \mathcal{B}_{[hm+i+j+k]_l,j}\end{aligned} \tag{18}$$

If we let $p=[k+hm]_l$, for $0\le i<m, 0\le j<n$, and $0\le h<t$, we have

$$\begin{aligned}\omega^{p}_{m\times n}\circ\tau(\mathcal{B})_{i,j} &= \tau(\mathcal{B})_{[i+p]_l,j}\\ &= \mathcal{B}_{[i+k+hm+j]_l,j}\end{aligned} \tag{19}$$

Since $0\le h<t$, there are total t sub-matrices in $\epsilon^{k}_{tm\times n}(\sigma(\bar{\mathcal{A}}))$ and $\omega^{k}_{tm\times n}(\tau(\mathcal{B}))$, the conclusion for the first part of the theorem follows naturally from Eqs. (16) to (19).

To prove the second part of the theorem, we only need to note that since $t=\left\lceil\frac{l}{m}\right\rceil$, we have $tm\ge l$. Therefore, for any $p\in\{0,1,\ldots,(l-1)\}$, we must be able to find at least one set of k and h, with $0\le k<m$, $0\le h<t$, and $p=[k+hm]_l$. Together with Eqs. (16) to (19), we thus prove the theorem. □

Theorem 3.6 *Let $\mathcal{A}_{m\times l}$ and $\mathcal{B}_{l\times n}$ with $n<l$, and let $\bar{\mathcal{B}}$ be matrix expanded with $t=\left\lceil\frac{l}{n}\right\rceil$ copies of $\mathcal{B}$ horizontally, i.e., $\bar{\mathcal{B}}=\{\mathcal{B};\mathcal{B};\ldots;\mathcal{B}\}$. Then,*

- $\epsilon^{k}_{m\times tn}(\sigma(\mathcal{A}))\odot\omega^{k}_{m\times tn}(\tau(\bar{\mathcal{B}}))$ contains t items of $\epsilon^{p}_{m\times n}(\sigma(\mathcal{A}))\odot\omega^{p}_{m\times n}(\tau(\mathcal{B}))$, with $p=[k]_l,[k+n]_l,\ldots,[k+(t-1)n]_l$;
- $\epsilon^{k}_{m\times tn}(\sigma(\mathcal{A}))\odot\omega^{k}_{m\times tn}(\tau(\bar{\mathcal{B}}),\quad k=0,1,\ldots,(n-1)$ contains all items of $\epsilon^{p}_{m\times n}(\sigma(\mathcal{A}))\odot\omega^{p}_{m\times n}(\tau(\mathcal{B}))$, with $p=0,1,\ldots,(l-1)$.

Proof Consider a sub-matrix of $(\epsilon^{k}_{m\times tn}\circ\sigma(\mathcal{A}))$ with dimension of $m\times n$, i.e., $(\epsilon^{k}_{m\times tn}\circ\sigma(\mathcal{A}))_{i,hn+j}$, where $0\le i<m, 0\le j<n$. h is a constant with $0\le h<t$. Based on Eqs. (1) and (3), we have

$$\begin{aligned}(\epsilon^{k}_{m\times tn}\circ\sigma(\mathcal{A}))_{i,hn+j} &= \sigma(\mathcal{A})_{i,[hn+j+k]_l}\\ &= \mathcal{A}_{i,[i+hn+j+k]_l}\end{aligned} \tag{20}$$

On the other hand, let $p=[k+hn]_l$, for $0\le i<m, 0\le j<n$, we have

$$\begin{aligned}(\epsilon^{p}_{m\times n}\circ\sigma(\mathcal{A}))_{i,j} &= \sigma(\mathcal{A})_{i,[j+p]_l}\\ &= \mathcal{A}_{i,[i+j+k+hn]_l}.\end{aligned} \tag{21}$$

Similarly, consider the sub-matrix of $(\omega^k_{m\times tn}\circ\tau(\bar{\mathcal{B}}))$ with dimension of $m\times n$, i.e., $(\omega^k_{m\times tn}\circ\tau(\bar{\mathcal{B}}))_{i,hn+j}$, with $0\le i<m, 0\le j<n$. Based on Eq. (2) and (4), we have

$$\begin{aligned}(\omega^k_{m\times tn}\circ\tau(\bar{\mathcal{B}}))_{i,hn+j} &= \tau(\bar{\mathcal{B}})_{[i+k]_l,hn+j}\\ &= \bar{\mathcal{B}}_{[hn+i+j+k]_l,hn+j}\\ &= \mathcal{B}_{[hn+i+j+k]_l,j}\end{aligned} \tag{22}$$

If we let $p=[k+hn]_l$, for $0\le i<m, 0\le j<n$, and $0\le h<t$, we have

$$\begin{aligned}\omega^p_{m\times n}\circ\tau(\mathcal{B})_{i,j} &= \tau(\mathcal{B})_{[i+p]_l,j}\\ &= \mathcal{B}_{[i+k+hn+j]_l,j}\end{aligned} \tag{23}$$

Since $0\le h<t$, there are total t sub-matrices in $\epsilon^k_{m\times tn}(\sigma(\mathcal{A}))$ and $\omega^k_{m\times tn}(\tau(\mathcal{B}))$, the conclusion for the first part of the theorem follows naturally from Eqs. (20) to (23).

To prove the second part of the theorem, we only need to note that since $t=\left\lceil\frac{l}{n}\right\rceil$, we have $tn\ge l$. Therefore, for any $p\in\{0,1,\ldots,(l-1)\}$, we must be able to find at least one set of k and h, with $0\le k<m, 0\le h<t$, and $p=[k+hn]_l$. Together with Eqs. (20) to (23), we thus prove the theorem. □

Appendix 2: Meaning of symbolize

See Table 9.

Table 9 Meaning of symbolize

Symbolize	Meaning
$\mathcal{A}$	Left matrix for matrix multiplication
$\mathcal{B}$	Right matrix for matrix multiplication
m	The number of row of matrix $\mathcal{A}$
l	The number of column of matrix $\mathcal{A}$
	The number of row of matrix $\mathcal{B}$
n	The number of column of matrix $\mathcal{B}$
σ	The transformation that permute each row
τ	The transformation that permute each column
ϵ	The transformation that permute multiple columns
ω	The transformation that permute multiple rows
ct	The prefix of ciphertext
$\mathbf{U}$	Permutation matrix
$\odot$	Elementwise multiplication

Appendix 3: Some experiments results

See Table 10.

Table 10 Performance comparison

	$\mathcal{A}$	$\mathcal{B}$	HEGMM	HEGMM-En	E2DM-S	E2DM-R	Huang et al	HEGMM v.s. E2DM	HEGMM v.s Huang et al	HEGMM-En v.s. E2DM	HEGMM-En v.s. Huang et al
$m = \min(m, l, n)$	$\mathcal{A}_{25\times42}$	$\mathcal{B}_{42\times25}$	3686.65	2047.32	4089.43	2408.43	8330.00	0.65	2.26	1.18	4.07
	$\mathcal{A}_{11\times51}$	$\mathcal{B}_{51\times11}$	4509.41	960.37	5670.28	1190.08	12684.21	0.26	2.81	1.24	13.21
	$\mathcal{A}_{5\times22}$	$\mathcal{B}_{22\times16}$	1788.83	466.97	1905.73	593.08	3582.00	0.33	2.00	1.27	7.67
	$\mathcal{A}_{2\times51}$	$\mathcal{B}_{51\times28}$	5002.78	951.24	5690.84	1251.43	12674.97	0.25	2.53	1.32	13.32
	$\mathcal{A}_{6\times11}$	$\mathcal{B}_{11\times25}$	901.93	457.07	2356.30	620.58	1582.98	0.69	1.76	1.36	3.46
	$\mathcal{A}_{5\times17}$	$\mathcal{B}_{17\times47}$	1372.44	471.64	4687.23	824.32	3186.70	0.60	2.32	1.75	6.76
	$\mathcal{A}_{28\times31}$	$\mathcal{B}_{31\times58}$	2705.24	2618.99	5850.27	N/A	5992.41	2.16	2.22	2.23	2.29
	$\mathcal{A}_{17\times54}$	$\mathcal{B}_{54\times27}$	4731.17	1315.65	5397.93	N/A	11675.24	1.14	2.47	4.10	8.87
	$\mathcal{A}_{11\times11}$	$\mathcal{B}_{11\times63}$	1013.28	909.18	5497.60	N/A	2623.23	5.43	2.59	6.05	2.89
	$\mathcal{A}_{6\times7}$	$\mathcal{B}_{7\times63}$	549.20	475.05	5534.19	N/A	1365.76	10.08	2.49	11.65	2.87
$l = \min(m, l, n)$	$\mathcal{A}_{40\times17}$	$\mathcal{B}_{17\times52}$	1256.17	1418.09	5100.05	N/A	3973.80	4.06	3.16	3.60	2.80
	$\mathcal{A}_{28\times15}$	$\mathcal{B}_{15\times55}$	1262.29	1266.57	5610.51	N/A	3064.57	4.44	2.43	4.43	2.42
	$\mathcal{A}_{45\times12}$	$\mathcal{B}_{12\times32}$	989.30	993.98	4720.91	N/A	1342.67	4.77	1.36	4.75	1.35
	$\mathcal{A}_{22\times12}$	$\mathcal{B}_{12\times43}$	1005.18	896.69	4712.86	N/A	1724.43	4.69	1.72	5.26	1.92
	$\mathcal{A}_{11\times11}$	$\mathcal{B}_{11\times63}$	1013.28	909.18	5497.60	N/A	2608.89	5.43	2.57	6.05	2.87
	$\mathcal{A}_{61\times7}$	$\mathcal{B}_{7\times30}$	621.00	615.76	6226.67	N/A	1178.08	10.03	1.90	10.11	1.91
	$\mathcal{A}_{57\times6}$	$\mathcal{B}_{6\times36}$	466.87	521.44	6245.82	N/A	821.15	13.38	1.76	11.98	1.57
	$\mathcal{A}_{39\times2}$	$\mathcal{B}_{2\times29}$	129.72	124.79	3838.58	N/A	293.60	29.59	2.26	30.76	2.35
	$\mathcal{A}_{61\times2}$	$\mathcal{B}_{2\times33}$	124.37	125.55	6229.57	N/A	258.03	50.09	2.07	49.62	2.06
	$\mathcal{A}_{36\times1}$	$\mathcal{B}_{1\times13}$	38.48	45.53	3010.37	N/A	135.70	78.23	3.53	66.12	2.98

Table 10 (continued)

	$\mathcal{A}$	$\mathcal{B}$	HEGMM	HEGMM-En	E2DM-S	E2DM-R	Huang et al	HEGMM v.s. E2DM	HEGMM v.s Huang et al	HEGMM-En v.s. E2DM	HEGMM-En v.s. Huang et al
$n = \min(m, l, n)$	$\mathcal{A}_{49\times33}$	$\mathcal{B}_{33\times26}$	3049.29	2561.85	5177.28	2357.73	4325.55	1.70	1.42	2.02	1.69
	$\mathcal{A}_{48\times55}$	$\mathcal{B}_{55\times28}$	4951.12	2275.08	5632.18	2519.21	14294.01	1.14	2.89	2.48	6.28
	$\mathcal{A}_{37\times43}$	$\mathcal{B}_{43\times18}$	4001.91	1741.97	4711.63	583.76	10609.82	1.18	2.65	2.70	6.09
	$\mathcal{A}_{41\times41}$	$\mathcal{B}_{41\times13}$	3423.93	1322.51	4265.11	831.42	8802.08	1.24	2.57	3.21	6.66
	$\mathcal{A}_{61\times55}$	$\mathcal{B}_{55\times19}$	4807.63	1806.42	6251.98	574.30	13998.30	1.30	2.91	3.46	7.75
	$\mathcal{A}_{23\times17}$	$\mathcal{B}_{17\times6}$	1209.17	526.42	2050.58	463.54	1962.97	1.70	1.62	3.90	3.73
	$\mathcal{A}_{58\times50}$	$\mathcal{B}_{50\times13}$	4146.49	1287.49	5843.53	241.86	10212.88	1.41	2.46	4.54	7.93
	$\mathcal{A}_{31\times10}$	$\mathcal{B}_{10\times6}$	726.38	437.07	2718.88	5519.46	1080.04	3.74	1.49	6.22	2.47
	$\mathcal{A}_{40\times27}$	$\mathcal{B}_{27\times4}$	1903.22	467.37	3358.15	2500.17	5062.48	1.76	2.66	7.19	10.83
	$\mathcal{A}_{57\times29}$	$\mathcal{B}_{29\times4}$	2275.52	436.26	6101.99	5501.94	6049.75	2.68	2.66	13.99	13.87
$(m \times l) \times (l \times l) l\%m = 0$	$\mathcal{A}_{7\times21}$	$\mathcal{B}_{21\times21}$	1929.84	655.94	1917.67	673.63	3303.63	0.65	0.91	1.01	1.41
	$\mathcal{A}_{11\times22}$	$\mathcal{B}_{22\times22}$	1902.28	952.11	1886.15	936.67	3252.46	0.72	0.93	0.99	1.28
	$\mathcal{A}_{5\times5}$	$\mathcal{B}_{5\times5}$	334.38	344.40	334.00	346.14	400.11	1.00	0.70	0.98	0.69
	$\mathcal{A}_{3\times9}$	$\mathcal{B}_{9\times9}$	665.15	253.94	666.72	274.87	814.73	0.68	0.67	1.03	1.01
	$\mathcal{A}_{5\times20}$	$\mathcal{B}_{20\times20}$	1583.07	432.03	1564.49	458.62	2240.04	0.29	1.41	1.06	5.18
	$\mathcal{A}_{2\times6}$	$\mathcal{B}_{6\times6}$	390.98	164.37	376.71	164.62	475.58	0.67	0.69	1.00	1.03
	$\mathcal{A}_{2\times4}$	$\mathcal{B}_{4\times4}$	210.97	119.34	198.53	112.51	216.32	0.71	0.64	0.97	0.87
	$\mathcal{A}_{15\times45}$	$\mathcal{B}_{45\times45}$	4671.37	1556.49	4687.38	1563.12	9640.10	0.64	1.12	1.00	1.75
	$\mathcal{A}_{11\times22}$	$\mathcal{B}_{22\times22}$	1903.59	948.83	1885.01	935.70	3252.78	0.72	0.94	0.99	1.29
	$\mathcal{A}_{15\times60}$	$\mathcal{B}_{60\times60}$	5028.94	1284.90	5068.26	1290.52	10352.81	0.26	2.06	1.00	8.06

Table 10 (continued)

	$\mathcal{A}$	$\mathcal{B}$	HEGMM	HEGMM-En	E2DM-S	E2DM-R	Huang et al	HEGMM v.s. E2DM	HEGMM v.s Huang et al	HEGMM-En v.s. E2DM	HEGMM-En v.s. Huang et al
Square	$\mathcal{A}_{27\times27}$	$\mathcal{B}_{27\times27}$	2691.39	2709.45	2702.20	2685.34	5038.88	1.00	1.87	0.99	1.86
	$\mathcal{A}_{26\times26}$	$\mathcal{B}_{26\times26}$	2432.31	2431.40	2410.14	2423.52	4382.63	0.99	1.80	0.99	1.80
	$\mathcal{A}_{32\times32}$	$\mathcal{B}_{32\times32}$	2043.36	2043.76	2038.88	2026.27	2662.56	0.99	1.30	0.99	1.30
	$\mathcal{A}_{40\times40}$	$\mathcal{B}_{40\times40}$	3342.80	3355.83	3353.98	3331.22	5714.32	1.00	1.71	0.99	1.70
	$\mathcal{A}_{19\times19}$	$\mathcal{B}_{19\times19}$	1850.84	1840.68	1829.47	1847.79	3294.38	0.99	1.78	0.99	1.79
	$\mathcal{A}_{15\times15}$	$\mathcal{B}_{15\times15}$	1216.17	1220.62	1214.94	1239.17	2225.93	1.00	1.83	1.00	1.82
	$\mathcal{A}_{45\times45}$	$\mathcal{B}_{45\times45}$	4710.64	4745.30	4740.90	N/A	10404.80	1.01	2.21	1.00	2.19
	$\mathcal{A}_{29\times29}$	$\mathcal{B}_{29\times29}$	2951.67	2933.58	2932.04	N/A	5995.97	0.99	2.03	1.00	2.04
	$\mathcal{A}_{34\times34}$	$\mathcal{B}_{34\times34}$	2903.72	2888.51	2890.18	N/A	4584.24	1.00	1.58	1.00	1.59
	$\mathcal{A}_{63\times63}$	$\mathcal{B}_{63\times63}$	5477.50	5470.25	5476.33	N/A	13799.24	1.00	2.52	1.00	2.52

Acknowledgements This work was supported in part by the Air Force Office of Scientific Research (AFOSR) and the Air Force Research Laboratory/Information Directorate (AFRL/RI), Rome, NY under the 2021 Summer Faculty Fellowship Program, and Information Directorate Internship Program, respectively. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the US Government. Approved for Public Release on March 06, 2024. Distribution is Unlimited. Case Number: 2024-0184 (original case number(s): AFRL-2024-0944).

Author contributions YG and GQ wrote the main manuscript and YG prepared all figures and tables. LW reviewed and rewrote multiple parts of the manuscript. All authors reviewed the manuscript.

Funding This research was supported by funding from the Air Force Office of Scientific Research (AFOSR) and the Air Force Research Laboratory/Information Directorate (AFRL/RI), Rome, NY, under the 2021 Summer Faculty Fellowship Program, Grant Number 2024-0184 (original case number: AFRL-2024-0944). The funding body had no role in the design of the study, the collection, analysis, or interpretation of data. Additional support was provided by NSF grant 1952792, 2321572 and CNS-2348733.

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

Declarations

Conflict of interest No, I declare that the authors have no conflict of interest as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Ethical statements It is not applicable. This study did not involve human participants, personal data, or any procedures requiring ethical approval.

References

1. Varghese B, Buyya R (2018) Next generation cloud computing: new trends and research directions. Futur Gener Comput Syst 79:849–861
2. Vasiljeva T, Shaikhulina S, Kreslins K (2017) Cloud computing: business perspectives, benefits and challenges for small and medium enterprises (case of Latvia). Procedia Eng 178:443–451
3. Scale R (2015) State of the cloud report. Technical report
4. Rajaraman V (2014) Cloud computing. Resonance 19(3):242–258
5. Rivest RL, Adleman L, Dertouzos ML et al (1978) On data banks and privacy homomorphisms. Found Secure Comput 4(11):169–180
6. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, pp 169–178
7. Brakerski Z, Gentry C, Vaikuntanathan V (2014) (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans Comput Theory (TOCT) 6(3):1–36
8. Ran R, Xu N, Wang W, Gang Q, Yin J, Wen W (2022) Cryptogcn: fast and scalable homomorphically encrypted graph convolutional network inference. Preprint arXiv:2209.11904
9. Smart NP, Vercauteren F (2014) Fully homomorphic SIMD operations. Des Codes Crypt 71(1):57–81
10. Ibarrondo A, Viand A (2021) Pyfhel: Python for homomorphic encryption libraries. In: Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp 11–16
11. Fan J, Vercauteren F (2012) Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive
12. Brakerski Z (2012) Fully homomorphic encryption without modulus switching from classical GapSVP. In: Annual Cryptology Conference. Springer, pp 868–886
13. Cheon JH, Kim A, Kim M, Song Y (2017) Homomorphic encryption for arithmetic of approximate numbers. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer, pp 409–437

14. Ames S, Venkitasubramaniam M, Page A, Kocabas O, Soyata T (2020) Secure health monitoring in the cloud using homomorphic encryption: a branching-program formulation, pp 56–92. https://doi.org/10.4018/978-1-5225-9863-3.ch004
15. Nocker M, Drexel D, Rader M, Montuoro A, Schöttle P (2023) He-man–homomorphically encrypted machine learning with ONNX models. Preprint arXiv:2302.08260
16. Reagen B, Choi W-S, Ko Y, Lee VT, Lee H-HS, Wei G-Y, Brooks D (2021) Cheetah: optimizing and accelerating homomorphic encryption for private inference. In: IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE, pp 26–39
17. Masliah I, Abdelfattah A, Haidar A, Tomov S, Baboulin M, Falcou J, Dongarra J (2019) Algorithms and optimization techniques for high-performance matrix-matrix multiplications of very small matrices. Parallel Comput 81:1–21
18. Nagasaka Y, Matsuoka S, Azad A, Buluç A (2018) High-performance sparse matrix-matrix products on intel KNL and multicore architectures. In: Proceedings of the 47th International Conference on Parallel Processing Companion, pp 1–10
19. Jiang P, Hong C, Agrawal G (2020) A novel data transformation and execution strategy for accelerating sparse matrix multiplication on GPUs. In: Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp 376–388
20. Liu W, Vinter B (2014) An efficient GPU general sparse matrix-matrix multiplication for irregular data. In: IEEE 28th International Parallel and Distributed Processing Symposium. IEEE, pp 370–381
21. Valero-Lara P, Martínez-Pérez I, Mateo S, Sirvent R, Beltran V, Martorell X, Labarta J (2018) Variable batched DGEMM. In: 2018 26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp 363–367. https://doi.org/10.1109/PDP2018.2018.00065
22. Zhang Z, Wang H, Han S, Dally WJ (2020) SpArch: efficient architecture for sparse matrix multiplication. In: 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, pp 261–274
23. Lu W-j, Kawasaki S, Sakuma J (2016) Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data. Cryptology ePrint Archive
24. Halevi S, Shoup V (2014) Algorithms in HElib. In: Annual Cryptology Conference. Springer, pp 554–571
25. Duong DH, Mishra PK, Yasuda M (2017) 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
26. Mishra PK, Duong DH, Yasuda M (2017) Enhancement for secure multiple matrix multiplications over ring-LWE homomorphic encryption. In: Information Security Practice and Experience: 13th International Conference, ISPEC 2017, Melbourne, Proceedings 13. Springer, pp 320–330
27. Jiang X, Kim M, Lauter K, Song Y (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
28. Huang Z, Hong C, Weng C, Lu W-J, Qu H (2023) More efficient secure matrix multiplication for unbalanced recommender systems. IEEE Trans Dependable Secure Comput 20(1):551–562. https://doi.org/10.1109/TDSC.2021.3139318
29. Rathee D, Mishra PK, Yasuda M (2018) Faster PCA and linear regression through hypercubes in HElib. In: Proceedings of the 2018 Workshop on Privacy in the Electronic Society, pp 42–53
30. Huang H, Zong H (2022) Secure matrix multiplication based on fully homomorphic encryption. J Supercomput 1–22
31. Lyubashevsky V, Peikert C, Regev O (2010) On ideal lattices and learning with errors over rings. In: 29th International Conference on the Theory and Applications of Cryptographic Techniques. Springer, pp 1–23

Authors and Affiliations

Yang Gao[1] · Gang Quan[2] · Soamar Homsi[3] · Wujie Wen[4] · Liqiang Wang[1]

✉ Liqiang Wang
liqiang.wang@ucf.edu

Yang Gao
yang.gao@ucf.edu

Gang Quan
gaquan@fiu.edu

Soamar Homsi
soamar.homsi@us.af.mil

Wujie Wen
wwen2@ncsu.edu

[1] Department of Computer Science, University of Central Florida, Orlando, FL 32816, USA

[2] Electrical and Computer Engineering Department, Florida International University, Miami, FL 33174, USA

[3] Information Warfare Division, Air Force Research Laboratory, Rome, NY 610101, USA

[4] Department of Computer Science, North Carolina State University, North Carolina, Raleigh, United States