2025 IEEE 24th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 979-8-3315-6532-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/Trustcom66490.2025.00184

An Efficient FHE-based Ciphertext Matrix Multiplication Algorithm

Yihan Wang[1] Xiangjun Xue[2] Jingyong Liang[3] Donghai Guan[1] Çetin Kaya Koç[1,4,5] ✉

yihanwang@nuaa.edu.cn xxue358@aucklanduni.ac.nz Jingyong.Liang@monash.edu dhguan@nuaa.edu.cn cetinkoc@ucsb.edu

[1]Nanjing University of Aeronautics and Astronautics, China.
[2]University of Auckland, New Zealand.
[3]Monash University, Australia.
[4]Düzce University, Türkiye.
[5]University of California Santa Barbara, USA.

***Abstract*—People have a hard time using cloud computing because of rules concerning privacy and security in fields like healthcare and banking. Fully Homomorphic Encryption (FHE) lets computers work with encrypted data, but it puts a lot of burden on them. This paper introduces SMMHE (Secure Matrix Multiplication with FHE), a novel element-wise approach that enhances SIMD parallelism in contemporary FHE frameworks to mitigate costly rotation operations. SMMHE is 3.98× faster than the most common FHE-based multiplication techniques, according to thorough testing. This big speedup, which cuts the difference in performance between encrypted and plaintext computation by a lot, makes it much easier to use privacy-preserving cloud applications in real life, like classifying medical images. For encrypted MNIST classification, the amortized duration of 26 ms per image is an excellent example of this.**

***Index Terms*—Privacy preservation, Fully homomorphic encryption, Matrix multiplication.**

I. Introduction

Cloud computing is affordable and scalable [32]; yet, security issues persist, especially in vital areas like healthcare and banking [7], [26]. Homomorphic Encryption (HE) has emerged as a formidable solution to mitigate security and privacy concerns related to outsourcing data and computation to unreliable third-party cloud providers [30], [9], [4]. Gentry's groundbreaking research resulted in fully homomorphic encryption (FHE), enabling mathematical operations on ciphertexts [9]. A major problem is that FHE methods can be much more expensive to run than their plaintext counterparts, which makes it hard to put them into practice [27].

In fields like banking and healthcare where privacy-preserving machine learning applications are very important, fast encrypted matrix multiplication is a must. The exorbitant expense of these operations under FHE continues to be the major constraint [13], [16]. This study tackles computational expenses via the SIMD paradigm [31], which facilitates the integration of many data inputs into a single ciphertext for simultaneous homomorphic operations, thereby offering a more efficient approach for encrypted matrix multiplication.

We present a new way to securely multiply matrices $\mathcal{A}_{m\times l}$ and $\mathcal{B}_{l\times n}$, using σ, ϵ^k, and ω^k transformations to reduce homomorphic rotations, which gives us $O(l)$ complexity; the SMMHE algorithm, which fully uses SIMD parallelism and only needs l homomorphic multiplications and $2l$ rotations; a thorough theoretical analysis that proves its correctness and security in the semi-honest model [11]; and empirical validation in real-world applications like medical image classification and financial risk prediction, showing that it is ready for cloud deployment.

II. Related Work

Algorithmic advancements designed to accelerate matrix multiplication have constituted a substantial area of investigation in high-performance computing (e.g., [22], [24], [16], [19], [24], [33]). But these methods cannot be used directly to improve processes within the FHE framework because computational primitives are much more expensive.

Encrypting each element independently is an easy approach to accomplish matrix multiplication (MM) with fully homomorphic encryption (FHE). This makes $m \times n$ ciphertexts and $m \times l \times n$ multiplicative operations, which need a lot of space and computing power. A common method changes MM into a matrix-vector product calculation using SIMD methods [12], [31], but it still needs $m + n$ ciphertexts and $m \times n$ encrypted multiplications [20]). The polynomial embedding framework [23] facilitates homomorphic matrix multiplication via safe vector inner products, although it is only applicable for singular tasks with predetermined dimensions.

Jiang et al. [17] introduced an FHE MM approach for square matrices with $O(d)$ complexity, later extended to specific rectangular cases ($l \leq d$ and $[d]_l = 0$) at increased cost. The blocking approach in [14] makes rectangular

2324-9013/25/$31.00 ©2025 IEEE
DOI 10.1109/Trustcom66490.2025.00184

operations better by dividing them up into square submatrices. However, it only works when the dimensions are correct. Rathee et al. [28] proposed the encryption of matrices into a two-dimensional hypercube configuration [12], therefore converting matrix multiplication into sequential matrix-vector operations. Huang et al. [13] subsequently incorporated arbitrary matrix multiplication problems into this framework, which we utilized for evaluation.

This paper primarily employs Huang-MM [13] and E2DM [17] as foundational baselines, while acknowledging substantial progress in the optimization of FHE schemes [2], [5], [10], the development of hardware accelerators [29], [25], and the establishment of efficient programming frameworks [15]. These improvements aim to lower the overall overhead of FHE primitives (such as bootstrapping), which will help all high-level algorithms. The purpose of SMMHE is to work at the algorithm level to reduce the number of expensive FHE operations that MM needs. As a result, SMMHE is an extra optimization that may be combined with future low-level improvements to get speed benefits that are multiplicative. This makes it a good algorithmic benchmark for future evaluations.

III. Preliminaries

TABLE I
Summary of key notations and operators.

Symbol	Type	Description
$\mathcal{A}, \mathcal{B}$	Matrix	Plaintext matrices.
$\mathbf{v}$	Vector	Column vector.
$\mathbf{v}^\mathsf{T}$	Vector	Row vector (transposed).
$a_{i,j}$	Scalar	Element at (i, j).
$\mathbf{a}_i^\mathsf{T}$	Vector	i-th row of $\mathcal{A}$.
$\odot$	Operator	Element-wise product.
$\lll k$, $\gg k$	Operator	Cyclic left/right shift by k.
$\times$	Operator	Standard matrix multiplication.
$ct.\mathcal{A}$	Ciphertext	Encrypted matrix $\mathcal{A}$.
$\mathbf{Enc}(\cdot)$	Function	Encryption function.
$\mathbf{Add}$	FHE Op.	Homomorphic addition.
$\mathbf{Mult}$	FHE Op.	Homomorphic multiplication.
$\mathbf{CMult}$	FHE Op.	Homomorphic const. mult.
$\mathbf{Rot}$	FHE Op.	Homomorphic slot rotation.
$\sigma(\mathcal{A})$	Transform	Left-shift row i by i positions.
ϵ^k	Transform	Extracts/tiles the k-th diagonal.
ω^k	Transform	Shifts columns up by k rows.

We define the key notations in Table I). Let $\odot$ represent the element-wise (Hadamard) product. Homomorphic operations are: $\oplus$ (Add), $\otimes$ (Mult), $\odot$ (CMult), and Rot (rotation). Matrices are in calligraphic style ($\mathcal{A}, \mathcal{B}$), vectors are bold lowercase ($\mathbf{v}$), and matrix elements are $a_{i,j}$. A cyclic left shift by k is $\mathbf{a} \ll k$, a right shift is $\mathbf{a} \gg k$.

A. System and Security Models

Our structure has a client $\mathcal{C}$ and a cloud server $\mathcal{S}$. $\mathcal{C}$ encrypts the input matrices $\mathcal{A}, \mathcal{B}$ using fully homomorphic encryption (FHE) and sends them to $\mathcal{S}$, it then multiplies the ciphertexts and sends back the encrypted product. This keeps data private while it is being calculated [13]. We employ the semi-honest (honest-but-curious) adversary model [11], in which the adversary $\mathcal{S}$ adheres to the protocol while attempting to extract information from encrypted data. People trust $\mathcal{C}$, but $\mathcal{S}$, which could be multi-tenant. Security keeps $\mathcal{S}$ from learning anything about $\mathcal{A}, \mathcal{B}$, or $\mathcal{C}$.

One of the best things about SMMHE is that it may run without knowing any data. This means that the order and number of homomorphic operations depend only on public parameters (m, l, n) and not on secret values. This makes it naturally safe from assaults that use timing as a side channel. The semi-honest model ignores active attackers and physical side-channels. To make security better, we need verifiable computation [1] or hardware-level protections [6]. This will be looked at in future research.

For a secure outsourcing computation framework to be valid, the following properties must be satisfied [8]:

- Correctness: Faithful execution yields $\mathcal{C} = \mathcal{A} \times \mathcal{B}$.
- Security: $\mathcal{S}$ cannot extract meaningful information.
- Efficiency: $\mathcal{C}$'s workload is substantially lower than local MM.

B. Fully Homomorphic Encryption and SIMD

Common FHE schemes (CKKS [5], BFV [3], BGV [4]) allow computations on encrypted data. SIMD processing [31] packs multiple data elements into a single ciphertext for parallel computation. For ciphertexts $ct_x = \mathrm{Enc}(x_0, \ldots, x_N)$ and $ct_y = \mathrm{Enc}(y_0, \ldots, y_N)$:

- Add: $ct_x \oplus ct_y = \mathrm{Enc}(x_0 + y_0, \ldots, x_N + y_N)$
- Mult: $ct_x \otimes ct_y = \mathrm{Enc}(x_0 \times y_0, \ldots, x_N \times y_N)$
- CMult: $ct_x \odot pt = \mathrm{Enc}(x_0 \times p_0, \ldots, x_N \times p_N)$
- Rot: $\mathrm{Rot}(ct_x, i) = \mathrm{Enc}(x_i, \ldots, x_N, x_0, \ldots, x_{i-1})$

FHE-Mult is around $600\times$ slower than multiplying plain text [29], and operations add noise. SIMD reduces operations through batched computation, enabling safe cloud outsourcing for applications such as secure statistical testing and privacy-preserving machine learning [2].

C. Linear Transformation Packing Technique

Ciphertext packing [31] combines values into one ciphertext for SIMD. $\mathrm{Rot}(ct; l)$ rotates slots by l positions (negative $-l$ equals $n - l$). Following [13], arbitrary linear transformations use diagonal masks and rotations: for a $N \times N$ matrix U with d_U nonzero diagonals, it requires $O(d_U)$ rotations.

We adopt the column-major encoding strategy [13], [21], encrypting each row $\mathbf{x}_i^T$ separately as $\mathcal{A} \in \mathbb{R}^{m \times l}$ and $\mathcal{B} \in$

$\mathbb{R}^{l\times n}$, we define key transformation operators inspired by [19]:

- $\sigma(\mathcal{A})$: Cyclic left-rotation of row i by i positions, as shown in figure 1
- $\epsilon^k_{m\times n}(\mathcal{A})$: Extract/tile k-th cyclic diagonal of $\sigma(\mathcal{A})$
- $\omega^k_{m\times n}(\mathcal{B})$: Pad/crop $\mathcal{B}$ to $m\times n$, then rotate columns up by k rows

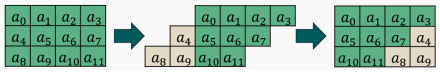

Fig. 1. The demonstration of σ operator.

Matrix multiplication is then expressed as:

$$\left(\epsilon^k \circ \sigma(\mathcal{A})\right) \odot \left(\omega^k \circ \tau(\mathcal{B})\right) \tag{2}$$

where $\tau(B)$ elevates column j by j rows.

where $\mathbf{u}^{(k)}_{\epsilon}$ is a binary mask for the k-th diagonal. Similarly for ω^k with mask $\mathbf{u}^{(k)}_{\omega}$, as shown in figure 2. The complete product is computed via mask-rotation steps.

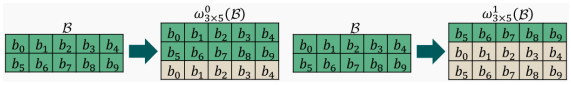

Fig. 2. For the matrix product $\mathcal{C}_{3\times5} = \mathcal{A}_{3\times2} \times \mathcal{B}_{2\times5}$ computed under encryption, we introduce two transformation matrices $\omega^0_{3\times5}(\mathcal{B})$ and $\omega^1_{3\times5}(\mathcal{B})$ to optimize homomorphic operations.

IV. Secure Matrix Multiplication Method

For SIMD-based fully homomorphic encryption (FHE) matrix multiplication to function well, the encrypted components must be precisely aligned to minimize costly slot rotations. We introduce SMMHE, a technique for safe matrix multiplication of any size that solely uses element-wise FHE operations. It uses the transformation operators σ, ϵ^k, and ω^k, as explained in Section III.

A. The SMMHE Algorithm

Given encrypted matrices $ct.\mathcal{A}_{m\times l}$ and $ct.\mathcal{B}_{l\times n}$, we compute the product via the element-wise formulation:

$$\mathcal{C} = \sum_{k=0}^{l-1} \epsilon^k(\sigma(\mathcal{A})) \odot \omega^k(\mathcal{B})$$

The computational complexity is dominated by the number of nonzero diagonals. We execute the following steps for encrypted matrices:

Step 1: Preprocessing

Apply the σ-transformation to $\mathcal{A}$, performing a cyclic left-shift of row i by i positions. Pad/duplicate $\mathcal{A}$ and $\mathcal{B}$ to common dimensions $m \times n$ using `generateUploadVector()`, handling four cases:

1) $m < l$ and $l > n$: Vertically extend $\mathcal{A}$ to $l \times n$ with zeros; $\widetilde{\mathcal{B}} = \mathcal{B}$.
2) $m < l < n$: Expand $\mathcal{A}$ vertically to $l \times n$ and duplicate horizontally; $\widetilde{\mathcal{B}} = \mathcal{B}$.
3) $m > l < n$: Duplicate $\mathcal{A}$ horizontally to $m \times n$; duplicate $\mathcal{B}$ vertically to $m \times n$.
4) $m > l > n$: $\widetilde{\mathcal{A}} = \mathcal{A}$; duplicate $\mathcal{B}$ vertically to $m \times n$.

Step 2: Encryption

Flatten matrices: $\hat{\mathcal{A}} \leftarrow \mathrm{vec}_F(\widetilde{\mathcal{A}})$, $\hat{\mathcal{B}} \leftarrow \mathrm{vec}_F(\widetilde{\mathcal{B}})$. Encrypt: $ct.\mathcal{A} \leftarrow \mathrm{Enc}(\hat{\mathcal{A}})$, $ct.\mathcal{B} \leftarrow \mathrm{Enc}(\hat{\mathcal{B}})$.

Step 3: Homomorphic Evaluation

For each $k = 0$ to $l-1$:

1) **Apply ϵ^k to extract the k-th diagonal:** Compute the encrypted k-th cyclic diagonal vector of $\sigma(\mathcal{A})$:
$$\mathrm{ct}.\mathcal{A}^{(k)} \leftarrow \mathrm{CMult}\left(\mathrm{Rot}\left(\mathrm{ct}.\mathcal{A}, r_k\right), \mathrm{mask}^k_{\epsilon}\right)$$
This step implements $\epsilon^k(\sigma(\mathcal{A}))$ and involves approximately l additions, $2l$ constant multiplications, and $3l$ slot rotations.
2) **Apply ω^k to shift matrix $\mathcal{B}$:** Compute the encrypted matrix $\mathcal{B}$ with each column shifted up by k rows:
$$\mathrm{ct}.\mathcal{B}^{(k)} \leftarrow \mathrm{CMult}\left(\mathrm{Rot}\left(\mathrm{ct}.\mathcal{B}, -s_k\right), \mathrm{mask}^k_{\omega}\right)$$
This step implements $\omega^k(\mathcal{B})$, requires l rotations (Rot) and l constant multiplications (CMult).
3) **Compute and accumulate the partial product:** Multiply the extracted diagonal from step 1 with the shifted matrix from step 2 and accumulate the result.
$$\mathrm{ct}.\mathcal{C} \leftarrow \mathrm{Add}\left(\mathrm{ct}.\mathcal{C}, \mathrm{Mult}\left(\mathrm{ct}.\mathcal{A}^{(k)}, \mathrm{ct}.\mathcal{B}^{(k)}\right)\right)$$
This involves l homomorphic multiplications (Mult) and $l-1$ Additions (Add).

Complexity Advantage: SMMHE achieves $O(l)$ complexity, a significant improvement over: Huang-MM [13] ($O(l^2)$), E2DM [17] ($O(l \log l)$), and a naive approach ($O(m \times l \times n)$).

B. Implementation Optimization

To enhance efficiency, we employ:

- **Mask Precomputation**: Generate $\mathbf{mask}^k_{\epsilon}$ and $\mathbf{mask}^k_{\omega}$ during initialization.
- **Rotation Scheduling**: Batch rotations to minimize key-switching operations.
- **Noise Management**: Dynamic modulus switching for multi-step computations.

Data-Oblivious Execution: The public matrix dimensions (m, l, n) set the order and number of homomorphic operations (additions, rotations, and multiplications). Secret values do not change this. This makes sure that patterns of memory access and runtime do not give away any information. This is naturally safe from timing-based side-channel attacks [18].

Figure 3 depicts the multiplication of matrix $\mathcal{A}_{5\times3}$ and $\mathcal{B}_{3\times4}$, with $m = 5$, $l = 3$, and $n = 4$.

Algorithm 1 SMMHE: Secure Matrix Multiplication

Require: Plaintext matrices $\mathcal{A}_{m\times l}$, $\mathcal{B}_{l\times n}$
Ensure: Ciphertext $ct.C$ encrypting $\mathcal{A}\times\mathcal{B}$

$\mathcal{A}_\sigma \leftarrow \sigma(\mathcal{A})$ ▷ Apply row-wise cyclic shift
$\hat{\mathcal{A}} \leftarrow \text{vec}_F(\widetilde{\mathcal{A}})$ ▷ Flatten column-major
$ct.A \leftarrow \mathbf{Enc}(\hat{A})$ ▷ Encrypt flattened $\widetilde{\mathcal{A}}$
$ct.B \leftarrow \mathbf{Enc}(\hat{B})$ ▷ Encrypt flattened $\widetilde{\mathcal{B}}$
$ct.C \leftarrow \mathbf{0}$ ▷ Initialize zero ciphertext
for $k = 0$ **to** $l-1$ **do**
  $rot_A \leftarrow \mathbf{Rot}(ct.A, k)$ ▷ Align k-th diagonal
  $ct_A^k \leftarrow \mathbf{CMult}(rot_A, \mathbf{mask}_\epsilon^k)$ ▷ Extract diagonal
  $rot_B \leftarrow \mathbf{Rot}(ct.B, -k)$ ▷ Inverse column shift
  $ct_B^k \leftarrow \mathbf{CMult}(rot_B, \mathbf{mask}_\omega^k)$ ▷ Column shift
  $ct_{\text{prod}} \leftarrow \mathbf{Mult}(ct_A^k, ct_B^k)$ ▷ Element-wise multiply
  $ct.C \leftarrow \mathbf{Add}(ct.C, ct_{\text{prod}})$ ▷ Accumulate result
  $\mathbf{ModSwitch}(ct.C)$ ▷ Noise management
end for
return $ct.C$

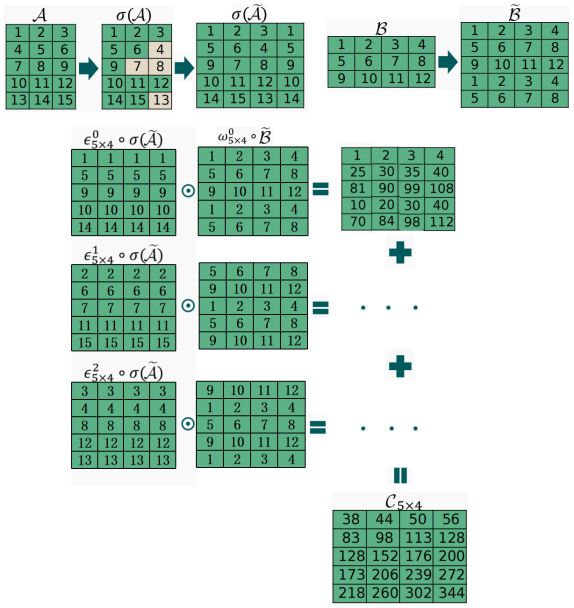

Fig. 3. SMMHE for $\mathcal{A}_{5\times3}\times\mathcal{B}_{3\times4}\rightarrow\mathcal{C}_{5\times4}$

V. Evaluation

We carefully compare SMMHE's performance to well-known FHE-based matrix multiplication methods, paying close attention to speed, memory use, and communication overhead, which are all important for cloud implementation.

A. Experimental Platform

SMMHE was implemented using Pyfhel [15], supporting the BFV [3]based on the RLWE hardness assumption [21]. Parameters: cyclotomic ring $R_q = \mathbb{Z}_q[X]/(\Phi_{4096}(X))$ with $\Phi_{4096}(X) = X^{2048}+1$, ring dimension $N = 4096$ (4096 plaintext slots), achieving NIST Level 1 security ($\geq$128-bit). We compare against **Huang-MM** [13] and **E2DM** [17] (both variants: E2DM-S for square, E2DM-R for rectangular). Tests used $2,000$ random matrix pairs with dimensions (m, l, n) sampled uniformly from $[1, 64]$ and 16-bit integer elements. All trials ran on an Ubuntu 22.04 server with dual Intel Xeon Silver 4114 CPUs (2.2 GHz, 10 cores/20 threads each) and 96 GB RAM. Preprocessing (σ-transformation for SMMHE, η-transformation for E2DM, diagonal extraction for Huang-MM) was done in plaintext before encryption. We measured end-to-end latency (encryption + computation + decryption).

B. Computational Time Analysis

Test instances were classified into five dimension groups:

1) Minimized row dimension ($m < l$ and $m < n$)
2) Minimized inner dimension ($l < m$ and $l < n$)
3) Minimized column dimension ($n < m$ and $n < l$)
4) Aligned dimensions ($l \mod m \equiv 0$)
5) Square matrices ($m = l = n$).

For E2DM, we report the faster result between E2DM-S and E2DM-R per test case. E2DM performance was modeled based on its $O(d^2)$ complexity and published benchmarks (64×64: 600 ms for E2DM-S, 282 ms for E2DM-R), scaled proportionally to dimension d.

Table II shows all the timing results for all the groups and techniques, and Table III shows how much faster SMMHE is than other approaches by using speedup ratios.

Key Observations:

- **Consistent Superiority:** SMMHE has the shortest computation time for all dimension groups, with average speedups of 2.92× over E2DM and 1.80× over Huang-MM.
- **Optimal Performance:** For square matrices (Group 5), SMMHE attains its maximum speedup (3.98× vs. E2DM).
- **Dimension Sensitivity:** The $l = \min$ group shows SMMHE's strongest relative performance (3.34× vs E2DM), highlighting its optimization for cases with minimized inner dimension.
- **Efficiency Factors:** SMMHE's advantage stems from its $O(l)$ complexity and minimized FHE operations (e.g., a 64×64 matrix multiplication completes in $\sim$ 201 ms with SMMHE versus $\sim$ 380 ms with Huang-MM).

C. Analysis of Memory and Communication Overhead

In cloud systems, server memory use and the bandwidth for client-server communication are quite limited. Metrics:

- *Server Memory:* Peak number of ciphertexts resident in memory during computation.

TABLE II
COMPUTATION TIME COMPARISON ACROSS DIMENSION GROUPS

Group	Method	Average	Median	Max
(1) $m = \min$	SMMHE	99.3	94.5	182.1
	Huang-MM	178.1	168.4	489.2
	E2DM-S	289.6	273.2	626.7
	E2DM-R	214.6	148.4	779.7
(2) $\ell = \min$	SMMHE	130.5	134.7	233.6
	Huang-MM	240.7	249.1	489.1
	E2DM-S	396.8	411.2	531.8
	E2DM-R	476.1	462.5	969.3
(3) $n = \min$	SMMHE	105.6	105.6	171.0
	Huang-MM	190.7	190.8	488.2
	E2DM-S	311.3	311.4	529.4
	E2DM-R	237.7	171.2	790.8
(4) $\ell \mod m \equiv 0$	SMMHE	97.8	101.8	164.8
	Huang-MM	175.2	183.2	480.2
	E2DM-S	284.7	298.3	435.4
	E2DM-R	134.8	66.9	851.9
(5) $m = \ell = n$	SMMHE	92.0	82.5	112.5
	Huang-MM	163.6	144.4	488.4
	E2DM-S	264.9	232.0	612.8
	E2DM-R	468.1	407.8	547.4

TABLE III
SPEEDUP OF SMMHE VS. OTHERS

Group	SMMHE vs E2DM	SMMHE vs Huang
(1) $m = \min$	2.53 (avg)	1.79 (avg)
(2) $\ell = \min$	3.34	1.84
(3) $n = \min$	2.60	1.81
(4) $\ell \mod m \equiv 0$	2.14	1.79
(5) $m = \ell = n$	3.98	1.78

- *Communication Overhead:* Total data transferred:

$$\text{Total Data} = (\text{IC} + \text{OC}) \times \text{Size}_{\text{ciphertext}}$$

where IC and OC denote Input and Output Ciphertext respectively.

Ciphertext size is ∼94 KB for $N = 4096$. SMMHE requires a constant number of ciphertexts (one for $\widetilde{\mathcal{A}}$, one for $\widetilde{\mathcal{B}}$, one accumulator for $\mathcal{C}$), while Huang-MM and E2DM require counts proportional to matrix dimensions or padded sizes.

Table IV presents the measured memory consumption and communication overhead for multiplying matrices $\mathcal{A}_{64\times64}$ and $\mathcal{B}_{64\times64}$, a common operation in practical applications like machine learning inference.

TABLE IV
SERVER MEMORY CONSUMPTION AND COMMUNICATION OVERHEAD FOR $\mathcal{A}_{64\times64} \times \mathcal{B}_{64\times64}$.

Method	Memory (KB)	Communication (KB)	Ciphertext Count
E2DM-S	658	1128	7
E2DM-R	752	1316	8
Huang-MM	470	846	5
SMMHE	**282**	**564**	**3**

The scalability of this advantage is illustrated in Figure 4. Results demonstrate SMMHE's clear advantage: it reduces server memory by 1.7× vs. Huang-MM and 2.3-2.7× vs. E2DM variants. Communication overhead is halved compared to Huang-MM and up to 2.7× lower than E2DM.

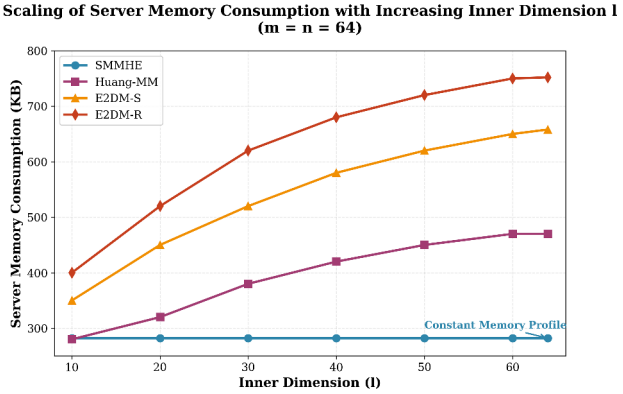

Fig. 4. Scaling of server memory consumption with increasing inner dimension l for $m = n = 64$.

Discussion: SMMHE is elegant since it is based on the FHE SIMD paradigm, hence reducing expensive data expansion and intricate ciphertext management. Less communication overhead is helpful when bandwidth is low, as it is in edge computing. The cloud costs less to run since it uses less memory and can do more encrypted calculations at once.

VI. CONCLUSIONS

This work thoroughly examined the substantial computational overhead linked to Fully Homomorphic Encryption (FHE)-based matrix multiplication, a pivotal constraint obstructing the practical implementation of privacy-preserving cloud services. We developed SMMHE (Secure Matrix Multiplication with FHE), an innovative technique that effectively utilizes Single Instruction Multiple Data (SIMD) parallelism using a distinctive element-wise formulation. This formulation combines the transformations of σ, ϵ^k, and ω^k to cut down on the number of expensive homomorphic rotations, which improves the overall speed of the computation.

SMMHE has a complexity of $O(l)$ with just l (the inner dimension of matrices $\mathcal{A} \in \mathbb{R}^{m\times l}$ and $\mathcal{B} \in \mathbb{R}^{l\times n}$) multiplications and $2l$ rotations. This makes it 3.98 times faster than other methods. This performance makes it possible to use things like encrypted medical picture categorization while still being safe under the semi-honest model and allowing data-oblivious execution. Future projects will include sparse matrices and hardware acceleration to make things more useful.

References

[1] Ben-Sasson, E., Chiesa, A., Tromer, E., and Virza, M. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology – CRYPTO* (2013), Springer, pp. 90–108.

[2] Boura, C., Gama, N., Georgieva, M., and Jetchev, D. Simulating homomorphic evaluation of deep learning predictions. In *International Symposium on Cyber Security Cryptography and Machine Learning: Third International Symposium, Beer-Sheva, Israel, June 27–28, 2019, Proceedings* (2019), Springer, pp. 212–230.

[3] Brakerski, Z. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference, Santa Barbara, CA, USA, 19-23 August, 2012* (2012), Springer, pp. 868–886.

[4] Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT) 6*, 3 (2014), 1–36.

[5] Cheon, J. H., Kim, A., Kim, M., and Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in cryptology–ASIACRYPT 2017: 23rd international conference on the theory and applications of cryptology and information security, Hong kong, China, December 3-7, 2017, proceedings, part I 23* (2017), Springer, pp. 409–437.

[6] Costan, V., and Devadas, S. Intel sgx explained. *Cryptology ePrint Archive*, 2016/086 (2016).

[7] Deeter, B. State of the cloud report. Tech. rep., Technical report, 2015.

[8] Gennaro, R., Gentry, C., and Parno, B. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30* (2010), Springer, pp. 465–482.

[9] Gentry, C. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June, 2009* (2009), Association for Computing Machinery, pp. 169–178.

[10] Gentry, C., Halevi, S., and Smart, N. P. Homomorphic evaluation of the aes circuit. In *Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2012* (2012), Springer, pp. 850–867.

[11] Goldreich, O. *Foundations of Cryptography, Volume II: Basic Applications*. Cambridge University Press, 2004.

[12] Halevi, S., and Shoup, V. Algorithms in helib. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34* (2014), Springer, pp. 554–571.

[13] Huang, H., and Zong, H. Secure matrix multiplication based on fully homomorphic encryption. *The Journal of Supercomputing 79*, 5 (2023), 5064–5085.

[14] Huang, Z., Hong, C., Weng, C., Lu, W.-j., and Qu, H. More efficient secure matrix multiplication for unbalanced recommender systems. *IEEE Transactions on Dependable and Secure Computing 20*, 1 (2021), 551–562.

[15] Ibarrondo, A., and Viand, A. Pyfhel: Python for homomorphic encryption libraries. In *CCS'21: 2021 ACM SIGSAC Conference on Computer and Communications Security Virtual Event, Republic of Korea, 15 November, 2021* (2021), Association for Computing Machinery, pp. 11–16.

[16] Jiang, P., Hong, C., and Agrawal, G. 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, San Diego, California, February 22–26, 2020* (2020), Association for Computing Machinery, pp. 376–388.

[17] Jiang, X., Kim, M., Lauter, K., and Song, Y. Secure outsourced matrix computation and application to neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, Canada, October 15–19, 2018* (2018), Association for Computing Machinery, pp. 1209–1222.

[18] Kocher, P. C. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology – CRYPTO '96* (1996), vol. 1109 of *Lecture Notes in Computer Science*, Springer, pp. 104–113.

[19] Liu, W., and Vinter, B. An efficient gpu general sparse matrix-matrix multiplication for irregular data. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, May 19-23, 2014* (2014), IEEE, pp. 370–381.

[20] Lu, W., Kawasaki, S., and Sakuma, J. Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data. In *Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 26 February–1 March, 2017* (2017), Internet Society, pp. 16–19.

[21] Lyubashevsky, V., Peikert, C., and Regev, O. On ideal lattices and learning with errors over rings. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29* (2010), Springer, pp. 1–23.

[22] Masliah, I., Abdelfattah, A., Haidar, A., Tomov, S., Baboulin, M., Falcou, J., and Dongarra, J. Algorithms and optimization techniques for high-performance matrix-matrix multiplications of very small matrices. *Parallel Computing 81* (2019), 1–21.

[23] Mishra, P. K., Duong, D. H., and Yasuda, M. Enhancement for secure multiple matrix multiplications over ring-lwe homomorphic encryption. In *Information Security Practice and Experience: 13th International Conference, ISPEC 2017, Melbourne, VIC, Australia, December 13–15, 2017, Proceedings 13* (2017), Springer, pp. 320–330.

[24] Nagasaka, Y., Matsuoka, S., Azad, A., and Buluç, A. High-performance sparse matrix-matrix products on intel knl and multicore architectures. In *Workshop Proceedings of the 47th International Conference on Parallel Processing, Eugene, OR, USA, August 13–16, 2018* (2018), Association for Computing Machinery, pp. 1–10.

[25] Nocker, M., Drexel, D., Rader, M., Montuoro, A., and Schöttle, P. HE-MAN–Homomorphically Encrypted MAchine learning with oNnx models. In *International Conference on Machine Learning Technologies, Stockholm, Sweden, March 10–March 12, 2023* (2023), Association for Computing Machinery, pp. 35–45.

[26] Rajaraman, V. Cloud computing. *Resonance 19* (2014), 242–258.

[27] Ran, R., Wang, W., Gang, Q., Yin, J., Xu, N., and Wen, W. Cryptogcn: Fast and scalable homomorphically encrypted graph convolutional network inference. In *NIPS'22: Proceedings of the 36th International Conference on Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December, 2022* (2022), no. 2731 in Advances in Neural Information Processing Systems, Curran Associates Inc., pp. 37676–37689.

[28] Rathee, D., Mishra, P. K., and Yasuda, M. Faster pca and linear regression through hypercubes in helib. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society, Toronto, Canada, 15 October, 2018* (2018), Association for Computing Machinery, pp. 42–53.

[29] Reagen, B., Choi, W.-S., Ko, Y., Lee, V. T., Lee, H.-H. S., Wei, G.-Y., and Brooks, D. Cheetah: Optimizing and accelerating homomorphic encryption for private inference. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, South Korea, 27 February–03 March, 2021* (2021), IEEE, pp. 26–39.

[30] Rivest, R. L., Adleman, L., Dertouzos, M. L., et al. On data banks and privacy homomorphisms. *Foundations of Secure Computation 4*, 11 (1978), 169–180.

[31] Smart, N. P., and Vercauteren, F. Fully homomorphic simd operations. *Designs, Codes and Cryptography 71* (2014), 57–81.

[32] Vasiljeva, T., Shaikhulina, S., and Kreslins, K. Cloud computing: Business perspectives, benefits and challenges for small and medium enterprises (case of latvia). *Procedia Engineering 178* (2017), 443–451.

[33] Zhang, Z., Wang, H., Han, S., and Dally, W. J. Sparch: Efficient architecture for sparse matrix multiplication. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), San Diego, CA, February 22-26, 2020* (2020), IEEE, pp. 261–274.