Euston: Efficient and User-Friendly Secure Transformer Inference with Non-Interactivity

Xinwen Gao[†], Shaojing Fu[†✉], Lin Liu[†✉], Zhuotao Liu[‡], Yuchuan Luo[†], Yongjun Wang[†]
[†]*National University of Defense Technology, Hunan, China*
[‡]*Tsinghua University, Beijing, China*

***Abstract*—Secure TransFormer Inference (STFI) frameworks have been proposed to address privacy concerns over user inputs and model parameters in Transformer-based LLMs. While most existing solutions rely on interactive protocols that incur substantial user-server communication overhead, non-interactive STFI variants have recently emerged to eliminate such dependencies. Nevertheless, state-of-the-art non-interactive STFI frameworks still suffer from critical limitations. (i) Large ciphertext sizes and multiple rotations alongside heavy user-side overhead in Homomorphic Matrix Multiplication (HMM). (ii) High approximation costs and depth consumptions in Homomorphic Nonlinear Evaluations (HNE).**

To address these limitations, we present Euston, an efficient and user-friendly STFI with non-interactivity. By combining RNS-CKKS fully homomorphic encryption with optimized methods, Euston achieves unprecedented efficiency in offline-online inference paradigm. The key innovations are twofold. (i) For linear operations, we adopt Singular Value Decomposition (SVD) with our novel batched HMMs to minimize ciphertext size and reduce rotation counts, simultaneously lowering user-side computational, communication and storage overhead. (ii) For nonlinear operations, we employ column(diagonal)-packed ciphertext matrix formats to eliminate costly rotations and depth regulation strategies to reduce depth consumption in non-interactive HNEs, which not only avoids user-server communications but also accelerates inference performance. In comparision with the state-of-the-art approach (NEXUS, NDSS 2025), Euston achieves up to 3100× lower preprocessing costs for the user and 8.8× higher system-wide inference performance, specifically delivering a 90× speedup for HMM and a 165.7× speedup for HNE. Our results demonstrate that Euston establishes new efficiency frontiers for user-friendly STFI deployment across cloud and edge environments.

1. Introduction

Advances in compute power and large-scale datasets have driven the rise of large language models (LLMs). The Transformer architecture breakthrough [1] introduced self-attention mechanisms and parallel computation, revolutionizing neural network scalability by enabling global sequence modeling. Nowadays, Transformer-based LLMs dominate modern AI applications through their architectural capacity to learn universal representations.

As the growing adoption of Transformer, researchers have delved into some critical privacy vulnerabilities in this paradigm [2], [3]. For instance, commercial LLM services like OpenAI's ChatGPT [4] safeguard their models as valuable proprietary assets, while utilizing user-submitted prompts for model refinement, inadvertently creating privacy leakage risks [5]. It is paramount to enable secure real-world deployment of Transformer without compromising input privacy or model confidentiality.

Secure TransFormer Inference (STFI) has emerged as a critical paradigm for mitigating privacy risks in Transformer-based service deployments. Recent studies [6], [7], [8], [9], [10], [11] demonstrate that cryptographic primitives, particularly fully homomorphic encryption (FHE) [12], [13], [14] and secure multi-party computation (MPC) [15], [16], [17], enable privacy-preserving evaluation of both linear and nonlinear operations during Transformer inference. Specifically, FHE supports arbitrary computations over encrypted data, covering the entire inference pipeline through Homomorphic Matrix Multiplications (HMM) and Homomorphic Nonlinear Evaluations (HNE). Meanwhile, MPC allows distributed parties to collaboratively execute operations without data exposure, often being hybridized with FHE to enhance efficiency and precision in nonlinear computations, which is a paradigm termed *interactive* STFI. However, even state-of-the-art interactive STFI frameworks like SIGMA [18] and BOLT [9] (alongside the previous works [7], [8], [19], [20]) incur prohibitive GB-level communication overhead for end users. To address this limitation, the recent approach NEXUS [10] implements both linear and nonlinear computations through fully FHE-based designs, pioneering the first *non-interactive* STFI framework that eliminates user-server communication. However, the design of non-interactive STFI still confronts two critical technical limitations that demand innovative solutions.

- **Inherent challenge and heavy user-side overhead in HMM.** The major challenge in HMM stems from the element-wise product formulation required by FHE schemes, which necessitates slot-unfilled ciphertexts and costly rotations. Therefore, the large-scale matrices in STFI usually incurs prohibitive communication and

✉*These authors contributed equally as co-corresponding authors. Email: {fushaojing, liulin16}@nudt.edu.cn*

computational overhead in HMM. While NEXUS provides an efficient SIMD-based secure compression and decompression algorithm, its requirement to process full ciphertext matrices still incurs larger ciphertext dimensions and higher runtime cost as model scales expand. Besides, we recognize that NEXUS omits the significant user-side overhead, which imposes hour-level computational costs and hundreds of MB storage footprints per user in the preprocessing phase. This is problematic for real-world deployment, as user devices typically operate under strict resource limitations.

- **Incompatible packing formats and high depth consumption in HNE.** We observe that the ciphertext matrix packing methodology critically determines the computational efficiency of HNE, particularly through its influence on required rotation operations. While NEXUS proposed SIMD slot folding algorithm, it defaults to compute HNE using row-packed ciphertext matrix, which still inherently demands extensive homomorphic rotations. Crucially, we recognize that NEXUS lacks explicit clarification about how to get such a format of ciphertext matrix through the preceding hidden-state Matrix Multiplication (HSMM) in their work. Besides, their adopted approximation methods for non-linear functions require substantial consumption of homomorphic multiplicative depth, directly amplifying the computational burden of bootstrapping operations.

1.1. Our Contributon

To address the above limitations, we present Euston, an **E**fficient and **u**ser-friendly **s**ecure **t**rans**fo**mer inference with **n**on-interactivity. Fully based on RNS-CKKS scheme, Euston enhance inference performance and systematically reduce user-side overhead, while ensuring input and weight confidentiality. At a high level, the user and server preprocess a mask matrix in the offline phase, and then perform HMM and HNE using the masked user input to finish STFI in the online phase.

Specifically, for HMM, Euston employs singular value decomposition (SVD) on the mask matrix and encrypts only the resulting diagonal singular value matrix, which significantly reduces transmitted ciphertext size. Since Euston involves many distinct matrix packing configurations, we further design four efficient HMM primitives through three fundamental computational perspectives: plaintext, ciphertext, and batched ciphertext. The first two primitives, tailored for column-packed, diagonal and unpacked matrices, eliminate redundant homomorphic rotations and mitigate user-side burdens during the Input-Weight Matrix Multiplication (IWMM) phase. The remaining two primitives, tailored for column-packed, row-packed and diagonal-packed matrices, fill the gap of HSMM by introducing component-wise ciphertext rotations. Moreover, we introduce column-packed and diagonal-packed ciphertext matrix formats that improve compatibility with downstream HNEs.

Subsequently, for HNE, we introduce some novel FHE-specific approximation methods and algorithms optimized for the proposed ciphertext formats. These innovations enable fully non-interactive computation by removing dependency on homomorphic rotations. Additionally, our depth regulation strategies reduce multiplicative depth requirements, thereby accelerating bootstrapping and improving end-to-end inference efficiency.

Our contributions are summerized as follows:

- **Efficient HMM primitives with user-friendliness.** We propose a novel secure and user-friendly matrix transmission protocol based on SVD and a suite of batched HMM primitives to execute linear operations in STFI. These novel methods achieve up to 90× faster HMM execution and 2.8× less communication bandwidth compared to NEXUS, while prioritizing user-side resource efficiency—BERT-Base inference requires 3100× shorter preprocessing runtime, 4.4× lower communication volume, and 2.2× less storage footprints.
- **Compatible HNE packing formats with lower depth consumption.** We propose some rotation-free and reduced-depth approximations for HNEs (GELU, Softmax, LayerNorm) by coordinating HMMs to preserve column-packed and diagonal-packed ciphertext matrix formats. Experimental validation confirms our methods attains up to a 17.7× GELU, 27.7× LayerNorm, and 165.7× Softmax acceleration relative to NEXUS, while enhancing inference performance by accelerating the bootstrapping phases by 1.25-1.3×.
- **Higher performance on end-to-end inference.** We implement Euston on both CPU and GPU architectures, demonstrating comprehensive performance superiority over NEXUS (NDSS 2025) [10] across all computational phases. End-to-end inference benchmarking reveals 3.5-8.8× CPU speedups and 2.1-3.7× GPU improvements across diverse model configurations. **Our source code is publicly available at https://github.com/FLL-Lab/Euston.**

2. Preliminaries

2.1. Notations

A comprehensive summary of frequent mathematical notations in this paper appears in Table 1. Specially, set notation is expressed using curly braces $\{\cdot\}$, with $\{\widehat{m}\}$ representing the integer range $\{1, 2, \ldots, m\}$.

2.2. Transformer Inference

Transformer [1] is a neural architecture based on self-attention mechanisms, forming the cornerstone of modern natural language processing. As illustrated in Fig. 1's server-side workflow, it employs stacked multi-head attention and feed-forward layers with LayerNorm to ensure stable inference. We formalize the core components as follows:

Attention Mechanism. The self-attention mechanism captures long-range dependencies by computing position-aware

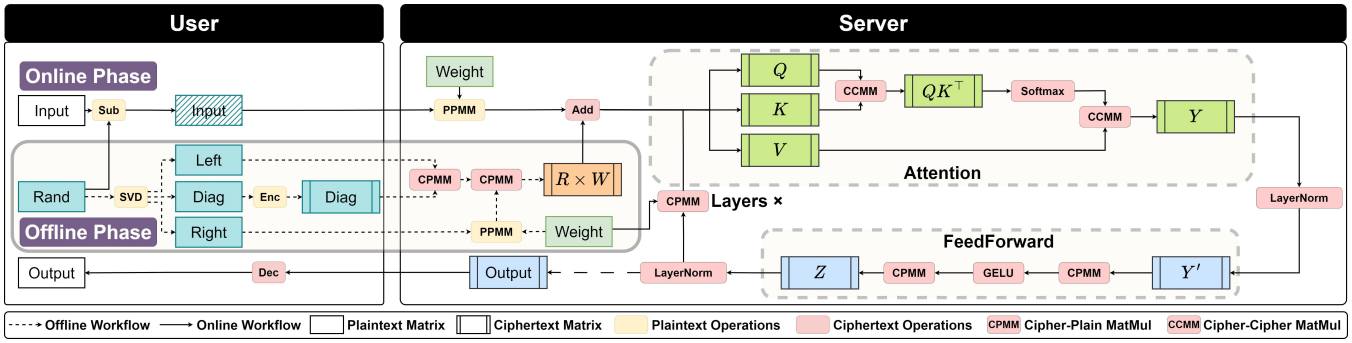


Figure 1: High-level view of Euston.

TABLE 1: Frequent Notations

Notations	Description
t	batchsize
m, n, b	dimensions
q	modulus
N	polynomial degree
$p(x)$	polynomial
$\mathbf{x} = (x_1, x_2, \ldots, x_m)$	vector
$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]$	matrix
$[\![\mathbf{x}]\!], [\![\mathbf{x}]\!]^{\leftarrow 1}, [\![\mathbf{x}]\!]^{\rightarrow 1}$	ciphertext and its rotations
$\widetilde{\mathbf{x}}, \widehat{\mathbf{x}}, \grave{\mathbf{x}}$	variants of vector
$\widehat{\mathbf{x}}^{\Leftarrow 1}, \widehat{\mathbf{x}}^{\Rightarrow 1}$	component-wise rotations

association weights across input sequences. Given an input embedding matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, linear projections generate query $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, key $\mathbf{K} = \mathbf{X}\mathbf{W}_K$, and value $\mathbf{V} = \mathbf{X}\mathbf{W}_V$ matrices. The attention operation computes:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{1}$$

where d_k denotes the key vector dimension. The scaling factor $\sqrt{d_k}$ mitigates gradient vanishing, while the Softmax normalization:

$$\text{Softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{m} \exp(z_j)}, \tag{2}$$

enhances both feature focus and numerical stability. The multi-head extension splits inputs into multiple parallel attention heads, concatenating their outputs through linear transformation.

Gaussian Error Linear Unit (GELU). GELU applies probabilistic gating for nonlinearity, with smoother gradients than ReLU to prevent neuron saturation in deep networks. Its analytical form is:

$$\text{GELU}(x) = \frac{1}{2}x\left(1 + \text{Erf}\left(\frac{x}{\sqrt{2}}\right)\right), \tag{3}$$

where $\text{Erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$ denotes the Gaussian error function.

Layer Normalization. LayerNorm stabilizes training by standardizing layer inputs. For $\mathbf{x} \in \mathbb{R}^n$, it computes:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}, \tag{4}$$

yielding normalized outputs:

$$y_i = \gamma \cdot \frac{x_i - \mu}{\sigma} + \beta, \tag{5}$$

where learnable parameters $\gamma, \beta \in \mathbb{R}^n$ restore representational capacity.

2.3. Fully Homomorphic Encryption (FHE)

FHE enables encrypted data computation while preserving privacy, forming the cryptographic basis for non-interactive security. We adopt the RNS-CKKS scheme [14] to implement Euston, with core algorithms specifically detailed as follows.

Encryption and Decryption. The scheme operates on the polynomial ring $\mathcal{R}_Q = \mathbb{Z}_Q[X]/(X^N + 1)$ with $Q = \prod_{i=0}^{L} q_i$. A plaintext vector $\mathbf{x} \in \mathbb{R}^{N/2}$ is encoded via canonical embedding into $m(X) = \pi(\mathbf{x}) \in \mathcal{R}_Q$, then encrypted using public key $\mathbf{pk}$:

$$[\![\mathbf{x}]\!] = \text{Enc}(m(X); \mathbf{pk}) \tag{6}$$

Decryption reconstructs $\mathbf{x}$ through ring projection and private key $\mathbf{sk}$:

$$\mathbf{x} = \pi^{-1}(\text{Dec}([\![\mathbf{x}]\!]; \mathbf{sk})) \tag{7}$$

Homomorphic Operations. RNS-CKKS supports element-wise arithmetic on ciphertexts. Given $[\![\mathbf{x}]\!]$ and $[\![\mathbf{y}]\!]$:

- Addition: $[\![\mathbf{x}]\!] \boxplus [\![\mathbf{y}]\!] = [\![\mathbf{x} + \mathbf{y}]\!]$,
- Multiplication: $[\![\mathbf{x}]\!] \boxtimes [\![\mathbf{y}]\!] = [\![\mathbf{x} \odot \mathbf{y}]\!]$,

where $\odot$ denotes Hadamard product. Each multiplication consumes one level from the depth budget L, necessitating bootstrapping when $L = 0$.

Rotation. Homomorphic rotation performs cyclic shifts over encrypted data. A left rotation by ω transforms $[\![\mathbf{x}]\!] = [x_0, \ldots, x_{N-1}]$ to $[\![\mathbf{x}]\!]^{\leftarrow\omega} = [x_\omega, \ldots, x_{N-1}, x_0, \ldots, x_{\omega-1}]$, realized via:

$$[\![\mathbf{x}]\!]^{\leftarrow\omega} \leftarrow \text{ROTATE}([\![\mathbf{x}]\!], \omega) \tag{8}$$

Rotation direction is encoded in ω's sign: positive values yield left shifts, negative values produce right rotations, satisfying:

$$[\![\mathbf{x}]\!]^{\rightarrow|\omega|} \equiv [\![\mathbf{x}]\!]^{\leftarrow\omega} \quad (\omega < 0) \tag{9}$$

Squaring. Homomorphic squaring is realized through self-multiplication:

$$\text{SQUARE}([\![\mathbf{x}]\!]) := [\![\mathbf{x}]\!] \boxtimes [\![\mathbf{x}]\!]. \tag{10}$$

Sign Function Approximation. We implement the sign function via composite polynomial approximation following prior work [10], [21], [22] (Δ normalizes inputs to $[-1, 1]$):

$$\text{SIGN}(x) \approx f^{d_f} \circ g^{d_g}(x/\Delta) \tag{11}$$

where basis polynomials f, g are iteratively applied d_f, d_g times. This enables encrypted sign evaluation:

$$\text{SIGN}([\![x]\!]) = \begin{cases} 1 & x > \epsilon \\ -1 & x \leq -\epsilon \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Bootstrapping. Bootstrapping [23] regenerates the ciphertext modulus as $Q' = Q \cdot q_{L+1}$via homomorphic decryption circuit evaluation, restoring the depth budget to $L' = L - K$, where K is the fixed recryption depth cost.

2.4. Singular Value Decomposition (SVD)

SVD decomposes a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ ($m \leq n$) into:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\top \tag{13}$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times m}$ are orthonormal, and $\mathbf{D} = \text{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{m\times m}$ contains ordered singular values $d_1 \geq \cdots \geq d_n \geq 0$. The factorization structurally organizes the matrix information: $\mathbf{U}$ and $\mathbf{V}$ encode orthogonal transformations, while $\mathbf{D}$ captures spectral features through singular values.

2.5. Secure Compression and Decompression

The secure compression and decompression algorithm proposed in NEXUS [10] is an efficient SIMD technique for packing ciphertexts. Specifically, the Secure Compression algorithm enables the direct encryption of N data points $\mathbf{a} = (a_0, a_1, \ldots, a_{N-1})$ as coefficients of a polynomial $f_{\mathbf{a}}(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{N-1}x^{N-1}$, which can then be homomorphically encrypted to produce a single ciphertext:

$$[\![f_{\mathbf{a}}(x)]\!] \leftarrow \text{SecComp}(\mathbf{a}). \tag{14}$$

Subsequently, the Secure Decompression algorithm can be used to obtain the encrypted replicated vectors $\widetilde{\mathbf{a}}_i = (a_i, a_i, \ldots, a_i)$ for $i \in \{0, 1, \ldots, N-1\}$, i.e.,

$$[\![\widetilde{\mathbf{a}}_i]\!] \leftarrow \text{SecDecomp}([\![f_{\mathbf{a}}(x)]\!], i), \tag{15}$$

In our work, we utilize this algorithm to reduce the communication overhead by minimizing the size of transmitted data.

3. Overview

3.1. System Model and Threat Model

Following prior works [8], [9], [10], we consider a typical STFI scenario in Euston where the user transmits input matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ to the server, and the server generate responses $\mathcal{M}(\mathbf{A})$ with the model $\mathcal{M}$. We assume both parties are semi-honest, which means that they execute protocols correctly but the server tries to learn $\mathbf{A}$ and the user tries to learn $\mathcal{M}$. Therefore, STFI must guarantee three security properties: 1) Protection of user inputs, 2) Protection of server model parameters, and 3) Correctness of inference output.

3.2. Problem Formulation

In semi-honest STFI scenario, Euston focuses on:

(1) How to minimize user-side preprocessing computational, communication, and storage burdens during STFI.

(2) How to design HMM for batched high-dimensional ciphertext matrices, ensuring optimization of performance and compatibility with nonlinear operators.

(3) How to develop efficient non-interactive methods for HNE.

3.3. Technical Overview

To address the aforementioned problems, we propose an SVD-based secure transmission protocol and a suit of optimizations for linear and nonlinear operations in Euston. Euston takes as input the embedding matrix generated via standard positional encoding of input tokens [1], following established practices in previous STFI architectures [8], [9], [10], [20]. As shown in Fig. 1, Euston subsequently operates through two distinct phases:

Offline Phase. The user generates a random matrix and performs SVD to obtain singular vector matrices and a diagonal singular value matrix. Then the singular value matrix is encrypted using SecComp algorithm (cf. Section 2.5), with the ciphertext result and plaintext singular vector matrices all transmitted to the server. The server then precomputes the multiplication results of the random matrix and weight matrices using the received components.

Online Phase. The user masks the input matrix using the random matrix as a one-time pad and transmits the obfuscated data. The server homomorphically combines these with the precomputed ciphertexts to excute HMMs and HNEs to produce the encrypted inference output, which is sent back to the user for decryption.

3.3.1. Homomorphic Matrix Multiplications (HMM). Tailored for decomposed matrices and HNE operators, we develop four novel HMMs in Euston:

- PCCDMM: Plaintext Column-packed Matrix $\times$ Ciphertext Diagonal Matrix (cf. Section 4.2.1).
- CCPUMM: Ciphertext Column-packed Matrix $\times$ Plaintext Unpacked Matrix (cf. Section 4.2.2).

- CCCRMM: Ciphertext Column-packed Matrix $\times$ Ciphertext Row-packed Matrix (cf. Section 4.2.3).
- CDCCMM: Ciphertext Diagonal-packed Matrix $\times$ Ciphertext Column-packed Matrix (cf. Section 4.2.3).

The first two of them are designed for the secure transmission protocol while accelerating IWMM operation using decomposed matrices. The other two fill the gap of HSMM by introducing column-packed and diagonal-packed formats while maintaining compatibility with subsequent efficient nonlinear functions.

3.3.2. Homomorphic Nonlinear Evaluations (HNE). In Euston, we propose some novel rotation-free and reduced-depth approximation methods for HNEs (e.g., GELU, LayerNorm and Softmax). These methods necessitate that all ciphertext input matrices adhere to either a column-packed or diagonal-packed layout. Crucially, these required packing configurations are exactly produced by our two distinct HSMMs (CCCRMM and CDCCMM).

4. Efficient Secure Matrix Multiplication

In this section, we first introduce an SVD-based secure transmission protocol which minimizes ciphertext size and reduce the user overhead. The amortized cost comparisons in Table 2 demonstrate our significant efficiency improvements. Subsequently, we introduce all HMMs in Euston which are performed with batched matrices.

4.1. User-Friendly Secure Matrix Transmission

Secure transmission of large matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$ (w.l.o.g. $m < n$) faces prohibitive FHE-ciphertext communication overhead. NEXUS [10] reduces transmission costs by secure compression and decompression algorithms. However, it encrypts the entire large matrix and the complexity grows polynomially with matrix dimensions.

Our key innovation lies in utilizing SVD to decompose a random matrix, which serves as a one-time pad for input obfuscation. Rather than encrypting and transmitting the entire matrix, we transmit only the encrypted singular value matrix—a diagonal matrix containing singular values. Table 2 illustrates that our methods significantly reduces the ciphertext size.

Following the offline-online paradigm [10], [19], Euston realizes user-friendliness by minimizing user-side computational, communication and storage overheads. The high-level workflow is visualized in Fig. 1.

4.1.1. Offline Phase. The user generates a random matrix $\mathbf{R} \in \mathbb{R}^{m\times n}$ and performs SVD decomposition to obtain singular vector matrices $\mathbf{U} \in \mathbb{R}^{m\times m}, \mathbf{H} \in \mathbb{R}^{m\times n}$ and a diagonal matrix $\mathbf{D} \in \mathbb{R}^{m\times m}$. We denote $\mathbf{d} = (d_1, d_2, \ldots, d_n)$ as the vector of diagonal singular values. The user then transmits plaintext matrices $\mathbf{U}, \mathbf{H}$ and ciphertext $[\![f_{\mathbf{d}}(x)]\!] \leftarrow \mathrm{SecComp}(\mathbf{d})$ to the server.

To realize IWMM operation, the server computes $[\![\mathbf{E}]\!] = \mathbf{U} \boxtimes [\![\mathbf{D}]\!]$ via PCCDMM (cf. Section 4.2.1). Subsequently, the server computes $[\![\mathbf{RW}]\!] = [\![\mathbf{E}]\!] \boxtimes \mathbf{HW}$ via CCPUMM (cf. Section 4.2.2), where $\mathbf{HW}$ is the plaintext matrix production of $\mathbf{H}$ and the weigth matrix $\mathbf{W}$.

TABLE 2: Amortized Computational Costs in Secure Input-Weight Matrix Multiplications (BERT-Base parameters: dimensions $m = 128$, $n = 768$, $b = 64$, batch size $t = 128$, and degree $N = 8192$)

Method	Ciphertext Size	Key Switches
Iron [8]	55.4 ($\frac{2\sqrt{mnb}}{\sqrt{N}}$)	–
Bumblebee [20]	13 ($\frac{m(n+b)}{N}$)	408 ($\frac{mb\log N}{2\sqrt{N}}$)
BOLT [9]	13 ($\frac{m(n+b)}{N}$)	96 ($\sqrt{\frac{m^2n^2b}{N^2}}$)
Nimbus [19]	6.4 ($\frac{b}{\lceil N/n \rceil}$)	–
NEXUS [10]	1 ($\frac{mb}{N} + \frac{nb}{Nt}$)	768 ($\frac{2nb-2}{t}$)
Euston	**0.016** ($\frac{m}{N}$)	**2** ($\frac{2m}{t}$)

4.1.2. Online Phase. The user masks the input matrix $\mathbf{A}$ through one-time pad matrix $\mathbf{R}$, producing the masking matrix transmitted to the server:

$$\mathbf{X} = \mathbf{A} - \mathbf{R} \in \mathbb{R}^{m\times n}. \tag{16}$$

The server then computes the ciphertext product:

$$[\![\mathbf{AW}]\!] = [\![\mathbf{RW}]\!] \boxplus \mathbf{XW} \tag{17}$$

This protocol enables the efficient, privacy-preserving computation of IWMM to obtain attention matrices $[\![\mathbf{Q}]\!]$, $[\![\mathbf{K}]\!]$, and $[\![\mathbf{V}]\!]$, which are further evaluated via CCCRMM and CDCCMM (cf. Section 4.2.3).

4.1.3. Security Analysis. As illustrated in Section 3.1, we consider the semi-honest threat model where a semi-honest adversary $\mathcal{A}$ controlling the server. We focus on the security of secure transmission protocol and omit the system-wide inference as the remaining processes are fully based on FHE, which is proved secure in [10]. A critical security requirement for our proposed IWMM process is to achieve *indistinguishability* (IND) for matrix $\mathbf{A}$: the adversary cannot distinguish between the real protocol execution and a simulation where $\mathbf{A}$ is replaced by a random matrix. Formally, we define MM-IND Security as follows:

Definition 1 (MM-IND Security)**.** *A matrix multiplication protocol* Π *achieves* ***MM-IND (Matrix Multiplication Indistinguishability-based) Security*** *if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that:*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A},\Pi}(1^\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\Pi}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda) \tag{18}$$

where $\mathsf{Exp}^{\mathsf{real}}$ *represents the real protocol and* $\mathsf{Exp}^{\mathsf{ideal}}$ *represents an ideal world.*

Our security analysis and the proof of MM-IND security for Euston's Π_{IWMM} rely on several well-established cryptographic assumptions and a fundamental lemma. We state these as follows:

Assumption 1 (FHE Security)**.** *The underlying FHE scheme used for* $[\![\mathbf{D}]\!]$ *satisfies IND-CPA security [24].*

Assumption 2 (One-Time Pad Security)**.** *The one-time pad (OTP) [25] encryption of* $\mathbf{X}$ *ensures that* $\mathbf{X} = \mathbf{A} - \mathbf{R}$ *is computationally indistinguishable from a uniformly random matrix of the same dimensions given a uniformly random matrix* $\mathbf{R}$*.*

Assumption 3 (Haar Distribution Indistinguishability)**.** *The Haar measure [26], [27] guarantees uniformity: Given an i.i.d uniformly random matrix* $\mathbf{U}$ *following Haar measure, its distribution possesses left and right invariance: for any orthogonal matrix* $\mathbf{Q}$*,* $\mathbf{QU}$ *and* $\mathbf{UQ}$ *are distributed identically to* $\mathbf{U}$*. This implies that Haar-distributed matrices with i.i.d. uniform entries are statistically indistinguishable from uniformly random matrices after orthogonal transformations.*

Lemma 1. *For a uniform random matrix* $\mathbf{R} \in \mathbb{R}^{m\times n}$ *with i.i.d. entries following haar measure, let* $\mathbf{R} = \mathbf{UDV}^\top$ *be its Singular Value Decomposition (SVD). Then:*

1) *The orthogonal matrices* $\mathbf{U}$ *and* $\mathbf{V}$ *are statistically independent of the singular value matrix* $\mathbf{D}$ *and of each other.*
2) *The distributions of the left and right singular vector matrices* $\mathbf{U}$ *and* $\mathbf{V}$ *follow the Haar measure on their respective orthogonal groups.*

Proof Sketch. The proof for this lemma is a standard result deferred from random matrix theory. A more detailed proof can be found in Appendix A. □

With these definitions, assumptions, and foundational lemma in place, we can now state the main security guarantee of our IWMM protocol.

Theorem 1 (MM-IND Security of IWMM Protocol)**.** *Let* Π_{IWMM} *be the IWMM protocol where the user sends* $\{\mathbf{U}, \mathbf{V}, [\![\mathbf{D}]\!], \mathbf{X} = \mathbf{A} - \mathbf{R}\}$ *to the server, with* $\mathbf{R} = \mathbf{UDV}^T$ *being a uniformly random matrix. Under the assumptions 1, 2, 3 and Lemma 1, the protocol* Π_{IWMM} *satisfies MM-IND security. Specifically, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that:*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A},\Pi_{\mathsf{IWMM}}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\Pi_{\mathsf{IWMM}}}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda). \tag{19}$$

Proof Sketch. We demonstrate the MM-IND security of Π_{IWMM} through a sequence of hybrid games, showing computational indistinguishability between the real protocol execution and an ideal experiment. A detailed, step-by-step proof can be found in Appendix B. □

4.2. Secure Matrix Multiplication

We systematically present our proposed HMMs through three fundamental computational perspectives: plaintext operations, standard ciphertext operations, and batched ciphertext operations. The batched computation examples of all four methods are visually demonstrated in Fig. 2.

4.2.1. Column-Packed Matrix × Diagonal Matrix. This PCCDMM operation is the first step to compute IWMM in the offline phase.

Plaintext Foundation. We first analyze the plaintext operation $\mathbf{E} = \mathbf{U} \cdot \mathbf{D}$ where $\mathbf{D}$ is a diagonal matrix. The computation reduces to element-wise multiplication between columns of $\mathbf{U}$ and corresponding diagonal entries. For example, consider $m = 2$, we have:

$$\begin{aligned}\mathbf{E} &= \left\{ \begin{bmatrix} u_{11} \\ u_{21} \end{bmatrix} \cdot d_{11} \begin{bmatrix} u_{12} \\ u_{22} \end{bmatrix} \cdot d_{22} \right\} \\ &= \left\{ \begin{bmatrix} u_{11} \\ u_{21} \end{bmatrix} \odot \begin{bmatrix} d_{11} \\ d_{11} \end{bmatrix} \begin{bmatrix} u_{12} \\ u_{22} \end{bmatrix} \odot \begin{bmatrix} d_{22} \\ d_{22} \end{bmatrix} \right\},\end{aligned} \tag{20}$$

where $\odot$ denotes element-wise product.

Ciphertext Multiplication. Given a diagonal matrix $\mathbf{D}$ represented as an encrypted diagonal vector $[\![f_{\mathbf{d}}(x)]\!]$, we employ the SecDecomp algorithm (cf. Section 2.5) to expand it into m replicated ciphertext vectors: $[\![\widetilde{\mathbf{d}}_i]\!] = (d_i, d_i, \ldots, d_i), i \in \{\widehat{m}\}$. The encrypted result $[\![\mathbf{E}]\!]$ is then constructed column-by-column through homomorphic multiplication:

$$[\![\mathbf{e}_i]\!] = \mathbf{u}_i \boxtimes [\![\widetilde{\mathbf{d}}_i]\!], \quad i \in \{\widehat{m}\} \tag{21}$$

where $\mathbf{u}_i$ denotes the i-th column of $\mathbf{U}$.

Batched Ciphertext Multiplication. Similar to NEXUS [10], we suppose the user and server need to generate t-batched response at a time. Therefore, to handle such batched matrix multiplications, specialized construction methods become essential. Consider t matrices via SVD:

$$\begin{aligned}\mathbf{R}_1 &= \mathbf{U}_1 \cdot \mathbf{D}_1 \cdot \mathbf{H}_1 \\ \mathbf{R}_2 &= \mathbf{U}_2 \cdot \mathbf{D}_2 \cdot \mathbf{H}_2 \\ &\cdots\cdots \\ \mathbf{R}_t &= \mathbf{U}_t \cdot \mathbf{D}_t \cdot \mathbf{H}_t\end{aligned} \tag{22}$$

We pack the batched matrices $\mathbf{U}_l \in \mathbb{R}^{m\times m}$ $(l \in \{\widehat{t}\})$ in column-wise format:

$$\begin{aligned}\widehat{\mathbf{U}} = \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \vdots \\ \mathbf{U}_t \end{bmatrix} &= \begin{bmatrix} \mathbf{u}_{11} & \mathbf{u}_{12} & \ldots & \mathbf{u}_{1m} \\ \mathbf{u}_{21} & \mathbf{u}_{22} & \ldots & \mathbf{u}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{u}_{t1} & \mathbf{u}_{t,2} & \ldots & \mathbf{u}_{tm} \end{bmatrix} \\ &:= \begin{bmatrix} \widehat{\mathbf{u}}_1 & \widehat{\mathbf{u}}_2 & \ldots & \widehat{\mathbf{u}}_m \end{bmatrix} \in \mathbb{R}^{tm\times m}\end{aligned} \tag{23}$$

For multiple diagonal matrices $\mathbf{D}_l \in \mathbb{R}^{m\times m}$ $(l \in \{\widehat{t}\})$, we first construct the batched matrix $\widehat{\mathbf{D}} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{D}_2 & \cdots & \mathbf{D}_m \end{bmatrix}^\top$. Let each diagonal vector be $\mathbf{d}_l = (d_{l,1}, d_{l,2}, \ldots d_{l,m})$. These vectors are concatenated into a single vector:

$$\widehat{\mathbf{d}} = \begin{bmatrix} \mathbf{d}_1 & \mathbf{d}_2 & \cdots & \mathbf{d}_t \end{bmatrix} \in \mathbb{R}^{tm} \tag{24}$$

To perform batched plaintext-ciphertext multiplication, we first apply the SecDecomp algorithm to $\widehat{\mathbf{d}}$, generating tm

replicated vectors $[\![\widetilde{\mathbf{d}}_{l,i}]\!] = (d_{l,i}, \ldots, d_{l,i})$ for $l \in \{\widehat{t}\}, i \in \{\widehat{m}\}$.

To compute the batched matrix product:

$$\widehat{\mathbf{E}} = \begin{bmatrix} \mathbf{U}_1\mathbf{D}_1 \\ \vdots \\ \mathbf{U}_t\mathbf{D}_t \end{bmatrix} \coloneqq \begin{bmatrix} \widehat{\mathbf{e}}_1 & \cdots & \widehat{\mathbf{e}}_m \end{bmatrix} \in \mathbb{R}^{tm \times m}, \tag{25}$$

we construct t distinct tm-dimensional bit masks vectors:

$$\begin{aligned} \mathbf{s}_1 &= (\underbrace{1, \ldots, 1}_{m}, 0, \ldots, 0) \\ \mathbf{s}_2 &= (\underbrace{0, \ldots, 0}_{m}, \underbrace{1, \ldots, 1}_{m}, 0, \ldots, 0) \\ \mathbf{s}_3 &= (\underbrace{0, \ldots, \ldots, 0}_{2m}, \underbrace{1, \ldots, 1}_{m}, 0, \ldots, 0) \\ &\cdots\cdots \\ \mathbf{s}_t &= (\underbrace{0, \ldots, \ldots, \ldots, 0}_{(t-1)m}, \underbrace{1, \ldots, 1}_{m}) \end{aligned} \tag{26}$$

These masks enable element-wise extraction of specific components from batched matrix samples. For instance, $\widetilde{\mathbf{d}}_{l,i} \odot \mathbf{s}_l$ isolates the replicated vector of the i-th diagonal element from the l-th matrix sample.

Finally, the encrypted column vectors are computed through homomorphic operations:

$$[\![\widehat{\mathbf{e}}_i]\!] = \widehat{\mathbf{u}}_i \boxtimes \sum_{l=1}^{t} ([\![\widetilde{\mathbf{d}}_{l,i}]\!] \boxtimes \mathbf{s}_l), \tag{27}$$

where $i \in \{\widehat{m}\}$ and for simplity, the summation operator denotes homomorphic aggregation of multiple ciphertexts.

4.2.2. Column-Packed Matrix × Unpacked Matrix. This CCPUMM operation is not only the second step of IWMM to compute $[\![\widehat{\mathbf{Q}}]\!], [\![\widehat{\mathbf{K}}]\!], [\![\widehat{\mathbf{V}}]\!] \in \mathbb{R}^{tm \times b}$ in the offline phase, but also the HSMM of ciphertext hidden-state matrix and plaintext weights.

Plaintext Foundation. To reconstruct the original matrix $\mathbf{R}$ prior to SVD, we compute $\mathbf{R} = \mathbf{EH}$ where the column-packed matrix $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m] \in \mathbb{R}^{m \times m}$ interacts with the unpacked matrix $\mathbf{H} \in \mathbb{R}^{m \times n}$. Given the unpacked nature of $\mathbf{H}$, we isolate each element h_{jk} into replicated vectors for homomorphic compatibility. Consider a minimal case with $m = n = 2$ and $\mathbf{R} \coloneqq [\mathbf{r}_1\ \mathbf{r}_2]$, we have:

$$\begin{aligned} \mathbf{r}_1 &= \begin{bmatrix} e_{11} \\ e_{21} \end{bmatrix} \odot \begin{bmatrix} h_{11} \\ h_{11} \end{bmatrix} + \begin{bmatrix} e_{12} \\ e_{22} \end{bmatrix} \odot \begin{bmatrix} h_{21} \\ h_{21} \end{bmatrix} \\ \mathbf{r}_2 &= \begin{bmatrix} e_{11} \\ e_{21} \end{bmatrix} \odot \begin{bmatrix} h_{12} \\ h_{12} \end{bmatrix} + \begin{bmatrix} e_{12} \\ e_{22} \end{bmatrix} \odot \begin{bmatrix} h_{22} \\ h_{22} \end{bmatrix} \end{aligned} \tag{28}$$

Ciphertext Multiplication. Now given the left-hand ciphertext matrix $[\![\mathbf{E}]\!] = \left[[\![\mathbf{e}_1]\!], [\![\mathbf{e}_2]\!], \ldots, [\![\mathbf{e}_m]\!]\right]$, we expand each element of the right-hand plaintext matrix $\mathbf{H}$ into replicated vectors $\widetilde{\mathbf{h}}_{ij} = (h_{ij}, h_{ij}, \ldots, h_{ij}) \in \mathbb{R}^m$, where $i \in \{\widehat{m}\}, j \in \{\widehat{n}\}$. The column-packed ciphertext matrix $[\![\mathbf{R}]\!]$ is then computed column-wise as:

$$[\![\mathbf{r}_j]\!] = \sum_{i=1}^{m} ([\![\mathbf{e}_i]\!] \boxtimes \widetilde{\mathbf{h}}_{ij}), \quad j \in \{\widehat{n}\} \tag{29}$$

Batched Ciphertext Multiplication. We column-pack the batched matrix $\widehat{\mathbf{H}} \coloneqq \begin{bmatrix} \widehat{\mathbf{h}}_1 & \widehat{\mathbf{h}}_2 & \ldots & \widehat{\mathbf{h}}_n \end{bmatrix} \in \mathbb{R}^{tm \times n}$ like Eq. (23). To compute:

$$\widehat{\mathbf{R}} = \begin{bmatrix} \mathbf{U}_1 \cdot \mathbf{D}_1 \cdot \mathbf{H}_1 \\ \mathbf{U}_2 \cdot \mathbf{D}_2 \cdot \mathbf{H}_2 \\ \vdots \\ \mathbf{U}_t \cdot \mathbf{D}_t \cdot \mathbf{H}_t \end{bmatrix} \coloneqq \begin{bmatrix} \widehat{\mathbf{r}}_1 & \widehat{\mathbf{r}}_2 & \cdots & \widehat{\mathbf{r}}_n \end{bmatrix} \in \mathbb{R}^{tm \times n}, \tag{30}$$

we expand $\widehat{\mathbf{H}}$ into $m \times n$ distinct tm-dimensional replicated vectors. For $i \in \{\widehat{m}\}, j \in \{\widehat{n}\}$, we define:

$$\begin{aligned} \widetilde{\widehat{\mathbf{h}}}_{ji} \coloneqq (&\underbrace{\widehat{h}_{j,i}, \ldots, \widehat{h}_{j,i}}_{m}, \underbrace{\widehat{h}_{j,m+i}, \ldots, \widehat{h}_{j,m+i}}_{m}, \\ &\underbrace{\widehat{h}_{j,2m+i}, \ldots, \widehat{h}_{j,2m+i}}_{m}, \ldots, \underbrace{\widehat{h}_{j,(t-1)m+i}, \ldots, \widehat{h}_{j,(t-1)m+i}}_{m}), \end{aligned} \tag{31}$$

where $\widehat{h}_{j,lm+i}$ corresponds to the $(lm+i)$-th element of vector $\widehat{\mathbf{h}}_j$, which is equivalent to $\mathbf{H}_{l,ij}$ (the element at row i and column j of $\mathbf{H}_l, l \in \{\widehat{t}\}$). We can then derive each column of $[\![\mathbf{R}]\!]$ as:

$$[\![\widehat{\mathbf{r}}_j]\!] = \sum_{i=1}^{m} \left([\![\widehat{\mathbf{e}}_i]\!] \boxtimes \widetilde{\widehat{\mathbf{h}}}_{ji} \right). \tag{32}$$

4.2.3. Diagonal-Packed Matrix × Column-Packed Matrix. To compute the attention score matrix $[\![\mathbf{P}]\!] \coloneqq [\![\mathbf{Q}]\!] \boxtimes [\![\mathbf{K}^\top]\!]$ in STFI, we adapt the CCCRMM from BOLT [9] with minimal modifications for batched format. CCCRMM will generate a diagonal-packed ciphertext matrix:

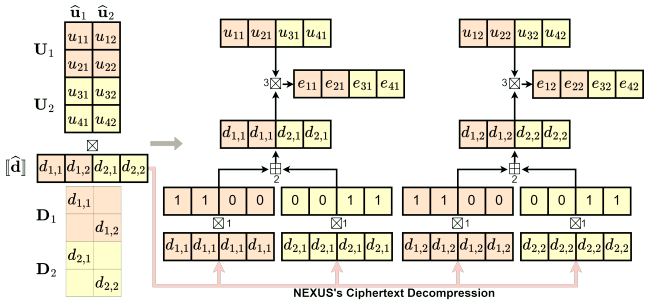
$$[\![\mathbf{P}]\!] \coloneqq [[\![\grave{\mathbf{p}}_1]\!], [\![\grave{\mathbf{p}}_2]\!], \ldots, [\![\grave{\mathbf{p}}_m]\!]], \tag{33}$$

where the technical details are illustrated in Appendix C.

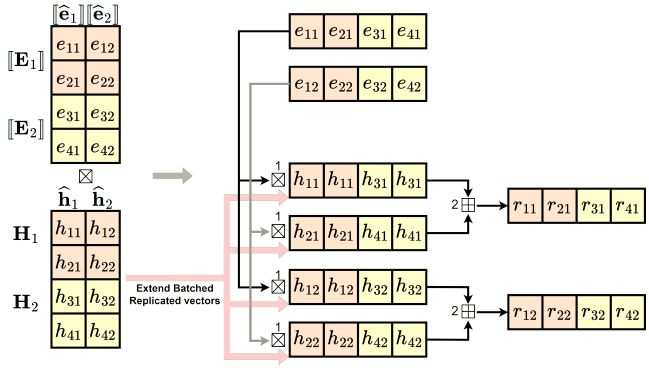
For the subsequent operation $[\![\mathbf{Y}]\!] \coloneqq [\![\mathbf{P}]\!] \cdot [\![\mathbf{V}]\!]$, we develop a novel CDCCMM between diagonal-packed $[\![\mathbf{P}]\!]$ and column-packed $[\![\mathbf{V}]\!]$. While BOLT provides a viable computation method for this multiplication operation, it produces MPC-optimized compact packing, which is incompatible with non-interactive nonlinear functions. Notably, the NEXUS framework entirely omits discussion of this multiplicative step, creating a foundational gap in its design. Therefore, we carefully design this operation and specifically generates column-packed outputs to facilitate the evaluation of nonlinear functions.

Plaintext Foundation. This matrix multiplication follows the Halevi-Shoup scheme [28]. Specifically, for a 2×2 matrix $\mathbf{G}$ and a 2-dimensional vector $\mathbf{c}$, the matrix-vector product is computed through:

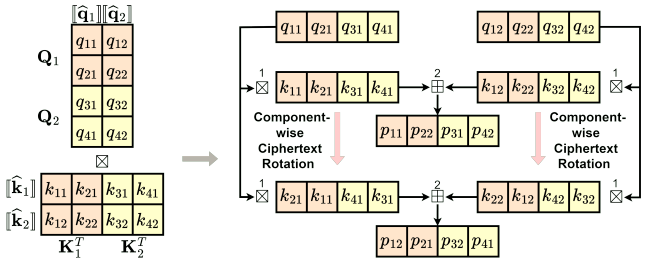
$$\begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} g_{11} \\ g_{22} \end{bmatrix} \odot \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} g_{12} \\ g_{21} \end{bmatrix} \odot \begin{bmatrix} c_2 \\ c_1 \end{bmatrix} \tag{34}$$



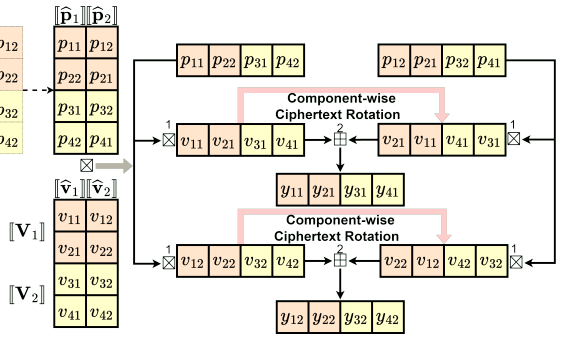
(a) PCCDMM: Plaintext Column-packed Matrix × Ciphertext Diagonal Vector.



(b) CCPUMM: Ciphertext Column-packed Matrix × Plaintext Unpacked Matrix.



(c) CCCRMM: Ciphertext Column-packed Matrix × Ciphertext Row-packed Matrix.



(d) CDCCMM: Ciphertext Diagonal-packed Matrix × Ciphertext Column-packed Matrix.

Figure 2: Examples of secure batched matrix multiplications in Euston. For simplicity, we set dimensions $m = 2, n = 2, b = 2$ and batch size $t = 2$. Additionally, the numbers next to the computation symbols clearly illustrate the order of operations.

Hence, each column of the matrix product between diagonal-packed and column-packed matrices can be computed independently.

Ciphertext Multiplication. Given a ciphertext diagonal-packed matrix $[\![\mathbf{P}]\!]$ and a ciphertext column-packed matrix $[\![\mathbf{V}]\!]$, each column of the output matrix $[\![\mathbf{Y}]\!] = \big[[\![\mathbf{y}_1]\!], [\![\mathbf{y}_2]\!], \ldots, [\![\mathbf{y}_b]\!]\big]$ can be computed as follows:

$$[\![\mathbf{y}_i]\!] = \sum_{j=1}^{m} ([\![\grave{\mathbf{p}}_j]\!] \odot [\![\mathbf{v}_i]\!]^{\leftarrow(j-1)}), \quad i \in \{\widehat{b}\} \tag{35}$$

Batched Ciphertext Multiplication. When performing ciphertext multiplication across a batch, it is infeasible to directly utilize the aforementioned approach for computation. The primary challange is that we need to achieve the inner rotation for each component of the batched column vector, instead of the whole vector. Taking the example of Fig. 2d, we need to transfor a batched vector $[\![\widehat{\mathbf{v}}_1]\!] = ([\![v_{11}]\!], [\![v_{21}]\!], [\![v_{31}]\!], [\![v_{41}]\!])$ to $[\![\widehat{\mathbf{v}}_1]\!]^{\Leftarrow 1} = ([\![v_{21}]\!], [\![v_{11}]\!], [\![v_{41}]\!], [\![v_{31}]\!])$ by performing the component-wise ciphertext rotation.

To achieve the desired rotation, we employ a bitmasking technique inspired by BOLT [9]. The complete procedure is formalized in Algorithm 1. Specifically, for a given rotation offset $\omega < m$, we construct two complementary bitmasks and batch t identical copies of each mask to match the ciphertext dimension. These masks are then Hadamard producted with both left-rotated and right-rotated versions of the column ciphertext vector $[\![\widehat{\mathbf{v}}]\!] = ([\![v_1]\!], [\![v_2]\!], \ldots, [\![v_{tm}]\!])$, respectively. Through the bitwise filtering mechanism, we efficiently identify the target bit positions and then aggregate the partial results via summation to produce the component-

Algorithm 1 Component-Wise Ciphertext Rotations

Input: Ciphertext $[\![\widehat{\mathbf{v}}]\!]$, batchsize t, and input rows m.
Output: A set of m component-wise rotated ciphertext.

1: **for** $\omega = 1, 2, \ldots, m$ **do**
2: $\quad \mathbf{s}_{\leftarrow\omega} \longleftarrow (\underbrace{1, 1, \ldots, 1}_{m-\omega}, \underbrace{0, 0, \ldots, 0}_{\omega})$
3: $\quad \widehat{\mathbf{s}}_{\leftarrow\omega} \longleftarrow (\underbrace{\mathbf{s}_{\leftarrow}, \mathbf{s}_{\leftarrow}, \ldots, \mathbf{s}_{\leftarrow}}_{t})$
$\quad = (\underbrace{1, \ldots,}_{m-\omega}\underbrace{0, \ldots,}_{\omega}\underbrace{1, \ldots,}_{m-\omega}\underbrace{0, \ldots,}_{\omega}\ldots,\underbrace{1, \ldots,}_{m-\omega}\underbrace{0, \ldots)}_{\omega}$
4: $\quad \widehat{\mathbf{s}}_{\rightarrow(\mathbf{m}-\omega)} \longleftarrow 1 - \widehat{\mathbf{s}}_{\leftarrow\omega}$
5: $\quad [\![\widehat{\mathbf{v}}]\!]^{\leftarrow\omega} \longleftarrow \text{ROTATE}([\![\widehat{\mathbf{v}}]\!], \omega)$
6: $\quad [\![\widehat{\mathbf{v}}]\!]^{\rightarrow(m-\omega)} \longleftarrow \text{ROTATE}([\![\widehat{\mathbf{v}}]\!], \omega - m)$
7: $\quad [\![\widehat{\mathbf{v}}]\!]^{\Leftarrow\omega} \longleftarrow [\![\widehat{\mathbf{v}}]\!]^{\leftarrow\omega} \boxtimes \widehat{\mathbf{s}}_{\leftarrow\omega} \boxplus [\![\widehat{\mathbf{v}}]\!]^{\rightarrow(m-\omega)} \boxtimes \widehat{\mathbf{s}}_{\rightarrow(\mathbf{m}-\omega)}$
8: **end for**
9: **return** $[\![\widehat{\mathbf{v}}]\!]^{\Leftarrow 1}, [\![\widehat{\mathbf{v}}]\!]^{\Leftarrow 2}, \ldots, [\![\widehat{\mathbf{v}}]\!]^{\Leftarrow m}$

wise rotated ciphertext.

$$[\![\widehat{\mathbf{v}}]\!]^{\Leftarrow\omega} = ([\![v_{\omega+1}]\!], [\![v_{\omega+2}]\!], \ldots, [\![v_m]\!], [\![v_1]\!], [\![v_2]\!], \ldots, [\![v_o]\!], [\![v_{m+\omega+1}]\!], \ldots, [\![v_{2m}]\!], [\![v_{m+1}]\!], \ldots, [\![v_{m+\omega}]\!], \ldots\ldots). \tag{36}$$

By using this algorithm on each batched column vector $[\![\widehat{\mathbf{v}}_i]\!]$ within the ciphertext matrix $[\![\widehat{\mathbf{V}}]\!]$, we can compute the corresponding column $[\![\widehat{\mathbf{y}}_j]\!]$ in the output matrix $[\![\widehat{\mathbf{Y}}]\!]$ as follows:

$$[\![\widehat{\mathbf{v}}_i]\!] = \sum_{j=1}^{m} ([\![\grave{\mathbf{p}}_j]\!] \boxtimes [\![\widehat{\mathbf{v}}_i]\!]^{\Leftarrow(j-1)}), \quad i \in \{\widehat{b}\}. \tag{37}$$

5. Efficient Secure Nonlinear Function

In this section, we introduce our novel approximations and algorithms for HNEs in non-interactive STFI, including Softmax, LayerNorm and GELU. As demonstrated in Section 4.2.3, all input matrices for nonlinear functions employ columnar or diagonal packing, which enables efficient rotation-free homomorphic approximation while maintaning the arithmetic accuracy in our Euston. For remaining nonlinear functions (e.g., Argmax), we directly adopt the established methods from NEXUS [10] and therefore omit detailed discussion in this work.

5.1. Approximate Arithmetic

HNEs require replacing exact arithmetic with approximation methods. Our designs focus on minimizing arithmetic multiplicative depth through algorithmic restructuring, thereby reducing bootstrapping overhead and inference runtime cost.

Polynomial Evaluation Polynomial approximation forms the foundation for HNEs. Given a polynomial $f(x) = \sum_{i=0}^{N} a_i x^i$, we optimize multiplicative depth through binary exponentiation [29]. This approach reduces the homomorphic multiplication depth while accelerating computation.

Exponential Function. For stable homomorphic evaluation of e^x in Softmax/GELU, we implement magnitude reduction via scaling decomposition [30]:

$$e^x = \Pi_{i=1}^{2^\tau} e^{x/2^\tau} = \underbrace{\left(\cdots\left(\left(e^{x/2^\tau}\right)^2\right)^2\cdots\right)^2}_{\tau \text{ squaring operations}}. \tag{38}$$

The scaled term $e^{x/2^\tau}$ with bounded input $x/2^\tau \in (-5, 0)$ is approximated by a 7-degree polynomial $f(x)$ minimizing $|f(x) - e^x|_\infty$. Empirical results demonstrate $\tau \leq 3$ achieves sufficient precision for transformer inference tasks. The complete procedure is formalized in Algorithm 2.

Multiplicative Inversion. HNEs universally require computing multiplicative inversions. We implement this through Goldschmidt's division algorithm, adopting NEXUS' optimized variant[1](see Algorithm 3) that constrains inputs to $x \in (0, 2)$. To satisfy this domain requirement, we apply pre-scaling and post-scaling transformations.

Algorithm 2 Homomorphic Exponential Approximation

Input: Ciphertext $[\![x]\!]$, polynomial $f(x) = \sum_{i=0}^{7} a_i x^i$, positive integer τ.
Output: $[\![y]\!] \approx \text{EXP}([\![x]\!])$

$[\![z]\!] \leftarrow [\![x]\!] \boxtimes \frac{1}{2^\tau}$
$[\![y]\!] \leftarrow f([\![z]\!])$ ▷ approximately EXP($[\![z]\!]$)
for $i = 1, 2, \ldots, \tau$ **do**
  $[\![y]\!] \leftarrow \text{SQUARE}([\![y]\!])$
end for
return $[\![y]\!]$

Algorithm 3 Homomorphic Inverse Approximation

Input: Ciphertext $[\![x]\!]$, scale γ, iteration θ.
Output: $[\![y]\!] \approx \text{INVERSE}([\![x]\!])$

$[\![x]\!] \leftarrow [\![x]\!] \boxtimes \gamma$
$[\![a]\!] \leftarrow \text{NEGATE}(-1 \boxplus [\![x]\!])$
$[\![b]\!] \leftarrow [\![a]\!] \boxplus 1$
$[\![y]\!] \leftarrow [\![b]\!]$
for $i = 1, 2, \ldots, \theta$ **do**
  $[\![a]\!] \leftarrow \text{SQUARE}([\![a]\!])$
  $[\![b]\!] \leftarrow [\![a]\!] \boxplus 1$
  $[\![y]\!] \leftarrow [\![y]\!] \boxtimes [\![b]\!]$
end for
$[\![y]\!] \leftarrow [\![y]\!] \boxtimes \gamma$
return $[\![y]\!]$

Square Root. The square root function is implemented through the third-order Taylor expansion of $\sqrt{1 + (x-1)}$:

$$sqrt(x) \approx 1 + \frac{1}{2}(x-1) - \frac{1}{8}(x-1)^2 + \frac{1}{16}(x-1)^3. \tag{39}$$

This approximation maintains numerical stability when $x \in (0, 2)$, achieved through scaling like multiplicative inversion.

5.2. Secure Nonlinear Functions

Nonlinear functions inherently need summation operations across all elements of each matrix row. When processing individual row vectors as inputs, the necessity for homomorphic rotations to perform row-wise summations imposes substantial computational overhead, particularly in batched scenarios.

To address this challenge, we adopt column-packed and diagonal-packed matrices as the standardized input format for HNEs. This design enables efficient summation through column-wise ciphertext additions, thereby significantly reducing computational overhead.[2] The secure algorithms for these functions are formally detailed in Algorithm 4, with comprehensive error evaluations under practical implementations provided in Table 3. Our optimization strategy

1. The NEXUS paper omits description of their customized Goldschmidt algorithm. We successfully reconstructed this optimized implementation through codebase examination.

2. The Softmax function typically operates on diagonal-packed inputs, its computational logic aligns with column-wise packing due to the mathematical equivalence between column-wise and diagonal vector summations.

Algorithm 4 Efficient Secure nonlinear Functions in Euston

Input: Column or diagonal-Packed Batched Ciphertext Matrix $[\![\widehat{\mathbf{X}}]\!] = \big[[\![\widehat{\mathbf{x}}_1]\!], [\![\widehat{\mathbf{x}}_2]\!], \ldots, [\![\widehat{\mathbf{x}}_m]\!]\big]$.

Output: Column or diagonal-Packed Batched Output $[\![\widehat{\mathbf{Y}}]\!]$.

Function: GELU ($[\![\widehat{\mathbf{X}}]\!]$)

for $i = 1, 2, \ldots, m$ **do**
  $[\![\widehat{\mathbf{s}}_i]\!] \longleftarrow \text{SIGN}([\![\widehat{\mathbf{x}}_i]\!])$
  $[\![\widehat{\mathbf{a}}_i]\!] \longleftarrow [\![\widehat{\mathbf{x}}_i]\!] \boxtimes [\![\widehat{\mathbf{s}}_i]\!]$
  $[\![\mathbf{z}_i]\!] \longleftarrow 1 \boxplus \text{EXP}(-1.702 \boxtimes [\![\mathbf{a}_i]\!])$
  $[\![\mathbf{z}_i]\!] \longleftarrow [\![\mathbf{a}_i]\!] \boxtimes \text{INVERSE}([\![\mathbf{z}_i]\!])$
  $[\![\widehat{\mathbf{b}}_i]\!] \longleftarrow (-1 \boxplus [\![\widehat{\mathbf{s}}_i]\!]) \boxtimes [\![\widehat{\mathbf{a}}_i]\!] \boxtimes 1.702/2$
  $[\![\widehat{\mathbf{c}}_i]\!] \longleftarrow [\![\widehat{\mathbf{s}}_i]\!] \boxtimes \text{EXP}([\![\widehat{\mathbf{b}}_i]\!])$
  $[\![\widehat{\mathbf{y}}_i]\!] \longleftarrow [\![\mathbf{z}_i]\!] \boxtimes [\![\widehat{\mathbf{c}}_i]\!]$
end for
Return $[\![\widehat{\mathbf{Y}}]\!] = \big[[\![\widehat{\mathbf{y}}_1]\!], [\![\widehat{\mathbf{y}}_2]\!], \ldots, [\![\widehat{\mathbf{y}}_m]\!]\big]$

Function: LayerNorm ($[\![\widehat{\mathbf{X}}]\!]$)

for i = 1,2,..., m **do**
  $[\![\widehat{\mathbf{a}}_i]\!] \longleftarrow \frac{1}{m} \boxtimes \sum_{j=1}^{m} ([\![\widehat{\mathbf{x}}_i]\!] \boxplus \text{NEGATE}([\![\widehat{\mathbf{x}}_j]\!]))$
end for
$[\![\widehat{\mathbf{b}}]\!] \longleftarrow \sum_{i=1}^{m} \text{SQUARE}([\![\widehat{\mathbf{a}}_i]\!])$
$[\![\widehat{\mathbf{b}}]\!] \longleftarrow \text{SQRT}(\lambda \boxtimes [\![\widehat{\mathbf{b}}]\!])$
$[\![\widehat{\mathbf{c}}]\!] \longleftarrow \text{INVERSE}([\![\widehat{\mathbf{b}}]\!])$
for i = 1,2,..., m **do**
  $[\![\widehat{\mathbf{y}}_i]\!] \longleftarrow (\gamma\sqrt{n\lambda} \boxtimes [\![\widehat{\mathbf{a}}_i]\!] \boxtimes [\![\widehat{\mathbf{c}}]\!])$
end for
Return $[\![\widehat{\mathbf{Y}}]\!] = \big[[\![\widehat{\mathbf{y}}_1]\!], [\![\widehat{\mathbf{y}}_2]\!], \ldots, [\![\widehat{\mathbf{y}}_m]\!]\big]$

Function: Softmax ($[\![\widehat{\mathbf{X}}]\!]$)

for i = 1,2,...,m **do**
  $[\![\widehat{\mathbf{a}}_i]\!] \longleftarrow \text{EXP}(-c \boxplus [\![\widehat{\mathbf{x}}_i]\!])$
end for
$[\![\widehat{\mathbf{a}}_{sum}]\!] \longleftarrow \sum_{i=1}^{m} [\![\widehat{\mathbf{a}}_i]\!]$
$[\![\widehat{\mathbf{b}}]\!] \longleftarrow \text{INVERSE}([\![\widehat{\mathbf{a}}_{sum}]\!])$
for i = 1,2,...,m **do**
  $[\![\widehat{\mathbf{y}}_i]\!] \longleftarrow [\![\widehat{\mathbf{a}}_i]\!] \boxtimes [\![\widehat{\mathbf{b}}]\!]$
end for
Return $[\![\widehat{\mathbf{Y}}]\!] = \big[[\![\widehat{\mathbf{y}}_1]\!], [\![\widehat{\mathbf{y}}_2]\!], \ldots, [\![\widehat{\mathbf{y}}_m]\!]\big]$

demonstrates particular efficacy in large-scale HNEs where rotational complexity dominates operational costs.

5.2.1. Batched GELU. Practical implementations of GELU function usually employ following two approximations [31]: Tanh Approximation:

$$\text{GELU}(x) \approx 0.5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right). \tag{40}$$

Sigmoid Approximation:

$$\text{GELU}(x) \approx x \cdot \sigma(1.702x), \quad \sigma(x) = \frac{1}{1+e^{-x}}. \tag{41}$$

Existing approaches [9], [10], [20] predominantly rely on Tanh approximations that demand three consecutive homomorphic sign evaluations and high-degree (≥ 6) polynomial approximations, inevitably accumulating excessive multiplicative depth. Instead, we pioneer the use of Sigmoid Approximation with a novel computational method, achieving equivalent functionality using only a single homomorphic sign operation while halving the required multiplicative depth.

To ensure numerical stability by constraining the input of the exponential function to negative values, we formulate the GELU approximation as:

$$y = x \cdot \sigma(1.702x) = \begin{cases} \frac{x \cdot \exp(1.702x)}{1+\exp(1.702x)}, & x \leq 0 \\ \frac{x}{1+\exp(-1.702x)}, & x > 0 \end{cases} \tag{42}$$

However, direct implementation remains challenging for column-packed batched ciphertext matrices $[\![\widehat{\mathbf{X}}]\!]$. Even when homomorphically determining the sign vector $\widehat{\mathbf{s}}_i$ for a column $[\![\widehat{\mathbf{x}}_i]\!]$, element-wise conditional computation is infeasible due to ciphertext packing constraints.

We therefore propose a sign-uniform transformation by computing the Hadamard product $\widehat{\mathbf{a}}_i = \widehat{\mathbf{x}}_i \odot \widehat{\mathbf{s}}_i$, ensuring $a = x \cdot s > 0$ for all elements. Then we can compute the unified term:

$$z_i = \frac{a}{1 + \exp(-1.702a)}. \tag{43}$$

To compute GELU, we can introduce auxiliary terms:

$$\begin{aligned} b &= \frac{s-1}{2} \cdot 1.702a = \begin{cases} -1.702a, & s = -1 \\ 0, & s = 1 \end{cases} \\ c &= s \cdot \exp(b) = \begin{cases} -\exp(1.702a), & s = -1 \\ 1, & s = 1 \end{cases} \end{aligned} \tag{44}$$

Finally, GELU preserves the piecewise definition through algebraic recombination:

$$y = z \cdot c = \begin{cases} \frac{-a \cdot \exp(-1.702a)}{1+\exp(-1.702a)}, & x \leq 0 \\ \frac{a}{1+\exp(-1.702a)}, & x > 0 \end{cases} \tag{45}$$

5.2.2. Batched Layer Normalization. To enhance compatibility with approximate arithmetic operations in homomorphic domains, we reformulate the original layer normalization by re-expressing the mean-adjusted term for row vectors in the input matrix:

$$a_i = x_i - \overline{x} = \frac{1}{n}\sum_{j=1}^{n}(x_i - x_j). \tag{46}$$

This permits the layer normalization output to be rewritten as:

$$\begin{aligned} y_i &= \gamma\frac{x_i - \overline{x}}{\sigma} + \beta \\ &= \gamma\frac{a_i}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} a_i^2}} + \beta \\ &= \gamma\frac{a_i\sqrt{n\lambda}}{\sqrt{(\lambda\sum_{i=1}^{n} a_i^2 - 1) + 1}} + \beta \end{aligned} \tag{47}$$

where scaling factor λ constrains $\sum_{i=1}^{n} a_i$ within $[0, 2]$ for square root approximations. In Euston's LayerNorm,

column-packed input matrices enable direct row summation through column vector addition, bypassing homomorphic rotations on individual row vectors.

5.2.3. Batched Softmax. The softmax operation requires row-wise element computation through:

$$y_i = \frac{\exp(x_i - c)}{\sum_{j=1}^{m} \exp(x_j - c)} \tag{48}$$

where $c > x_{\max}$ denotes an arbitrary substitution constant that cancels through division, eliminating explicit maximum value computation while maintaining numerical stability. Recall that Euston processes softmax using diagonal-packed ciphertext matrices $[\![\mathbf{X}]\!] = \left[[\![\hat{\mathbf{x}}_1]\!], [\![\hat{\mathbf{x}}_2]\!], \ldots, [\![\hat{\mathbf{x}}_m]\!]\right]$, which preserves row-wise summation capability through direct aggregation of packed vectors. Therefore, following exponential evaluation over diagonal vectors, we can compute the row-wise summation $[\![\mathbf{a}_{\text{sum}}]\!]$ via vector accumulation. The final softmax values are then obtained through multiplication of multiplicative inversion and each vector, which avoids the need for row-wise computaions and reduces the computanional overhead.

5.3. Depth Regulation

As RNS-CKKS operates under leveled homomorphic encryption with finite multiplicative depth capacity, Euston implements adaptive depth regulation to optimize computational efficiency while preventing ciphertext corruption. Our strategy focuses on two complementary approaches:

Scaling Factor Optimization. The default scaling factor $\lambda = 2^{40}$ in RNS-CKKS encoding enhances the precision of the decryption process. Therefore, for nonlinear operations like GELU (Line 4 in Algorithm 4), we can pre-adjust the scaling factor of $[\![\widehat{\mathbf{a}}_i]\!]$ to $\lambda \cdot 1.702^{-1}$, effectively implementing constant multiplication without consuming multiplicative depth. By doing this, the error induced by mutiplication can also be reduced to scaling error, as the bound of encryption noise $\mathcal{B}_{mult} > \mathcal{B}_{scale}$ [14].

Adaptive Parameter Selection. Euston's approximate arithmetic employs tunable parameters that affect multiplicative depth, such as squaring iterations τ in exponentials and iteration count θ in multiplicative inversion. Our adaptive adjustment enables Euston to select these values: smaller τ when exponential inputs are smaller (ensuring $x/2^{\tau} > -5$); reduced θ when inversion inputs approach 1.

Combined with binary exponentiation for polynomial evaluation (reducing depth from $\mathcal{O}(N)$ to $\mathcal{O}(\log N)$ for degree-N polynomials), these mechanisms achieve depth reduction and runtime optimization. Table 3 quantifies the multiplicative depth consumption across nonlinear operations, demonstrating Euston's superiority over prior framework.

5.4. Discussion

Our optimization for HNEs primarily benefits from our adoption of column-wise (diagonal) packed ciphertext matrices. To the best of our knowledge, this represents the first proposal of a vertical approach for approximating homomorphic matrix nonlinear functions. Compared to the conventional horizontal paradigm of row-wise packing, this method fundamentally eliminates the necessity for homomorphic rotations when computing slot-wise summations within a single ciphertext. Consequently, it not only enhances computational efficiency but also avoids the introduction of additional noise. This directly explains why Euston achieves smaller error than NEXUS—specifically, because the key switching involved in homomorphic rotation introduces noise with an upper bound of $\mathcal{B}_{ks} = 8\sigma N/\sqrt{3}$ [14], [23]. Moreover, we would also like to emphasize that the number of rows in the ciphertext matrix is of critical importance in our proposed HNE methods. A homomorphic matrix with fewer rows cannot be effectively computed using this approach due to significant slot underutilization within column-packed ciphertexts. Therefore, batching multiple matrices into a single matrix turns out to be a feasible solution to resolve this issue, which is exactly the case in HMMs of NEXUS [10] and our Euston.

TABLE 3: Depth consumption and experimental error of nonlinear functions.

Function	NEXUS		Euston	
	Depth	Error	Depth	Error
GELU	14	7.80E-03	12	3.70E-04
LayerNorm	16	5.00E-05	9	2.20E-05
Softmax	16	5.90E-02	10	2.50E-05

6. Performance Evaluation

6.1. Implementation

Building upon the NEXUS architecture, we have developed the Euston framework on both CPU and GPU platforms, implemented in C++ and extended to support a variety of models. Similar to NEXUS, Euston enables cross-platform deployment of the RNS-CKKS scheme by combining the CPU implementation inherited from Microsoft SEAL[3] with GPU acceleration through the Phantom framework[4] [32]. Adhering to the FHE standardization recommendations, we maintain consistency in core parameters: the polynomial ring dimension is set to $N = 2^{16}$, and a 1763-bit composite modulus is employed, providing 128-bit practical security. We construct a computational architecture with a multiplicative depth of 35 layers, of which 14 layers are specifically optimized for bootstrapping.

6.2. Experimental Setup

Our experimental evaluation adopts NEXUS as the primary baseline for comprehensive performance benchmarking. All CPU-based measurements were performed on an

3. https://github.com/Microsoft/SEAL
4. https://github.com/encryptorion-lab/phantom-fhe

AMD EPYC 7542 processor (2.90 GHz) with 128 GB RAM, running Ubuntu 22.04.1 LTS (Linux kernel 6.8.0-52-generic), where the thread number was fixed at 32 to maintain consistency with prior studies [8], [9], [10]. GPU tests utilized an NVIDIA RTX 6000 Ada Generation graphics card with 48 GB VRAM and CUDA 12.2 runtime environment.

When simulating network environments, we focus solely on a LAN setting, employing Linux Traffic Control to emulate a bandwidth of 3 Gbps and a round-trip latency of 0.8 ms. We omit simulations of WAN environments, as NEXUS has already demonstrated that non-interactive STFI incurs significantly lower communication overheads in WAN settings compared to MPC-based interactive STFI. In particular, during the assessment of nonlinear functions, the necessity for data transmission across WAN environments can be entirely eliminated. Moreover, the computational overhead in WAN environments is comparable to that in LAN environments.

In our experiments, all results are averaged over 10 runs. It is worth noting that since NEXUS only releases partially code for key algorithms, we cannot adjust the inputs and conduct experiments under the same scenarios for other algorithms. Therefore, for certain other metrics like model accuracy where device specifications introduce no measurement bias, we establish baseline comparisons through their published results. Additionally, we omit price[5] variation analysis in this study, as the our Euston-NEXUS comparison inherently reflects shifts in operational overheads, which directly align with AWS Financial Cloud's existing pricing framework.

6.3. Accuracy

To evaluate the inference accuracy of Euston, we assessed its performance on three datasets from the GLUE benchmark—RTE, SST-2, and QNLI—which are widely adopted as standard benchmarks for evaluating Transformer-based models. For these datasets, we employed BERT-Base and LLAMA-3-8B models fine-tuned specifically for each task and compared their inference accuracy under Euston with that of plaintext and NEXUS. As shown in Table 4, Euston's inference accuracy exhibits acceptable deviations from plaintext and consistently outperforms NEXUS across all tasks and models.

6.4. Microbenchmarks

This section presents the microbenchmarks evaluating matrix multiplication and nonlinear operations performance.

6.4.1. Matrix Multiplication. Adhering to the NEXUS benchmarking protocols, we evaluate this performance using BERT-Base model parameters. Fig. 3 compares the amortized runtime and communication overhead between Euston and NEXUS across varying batch sizes on both CPU and GPU platforms. Our analysis incorporates the aggregate costs across both offline and online phases for IWMM.

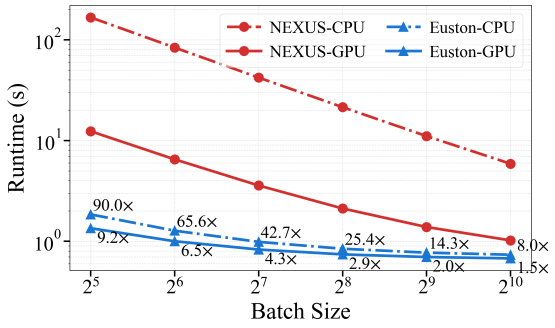
Key observations reveal significant efficiency gains for smaller batch configurations: Euston achieves 90× faster CPU computation and 9.2× GPU acceleration compared to NEXUS, while reducing communication volume by 2.8×. As batch sizes increase, the performance gap narrows due to fundamental limitations in online phase optimization—both frameworks share identical online computation patterns and communication requirements, establishing a lower bound for amortized cost reduction. We formally analyze this asymptotic behavior in Section 6.4.2, demonstrating how online phase characteristics ultimately constrain maximum achievable efficiency improvements.

Notably, Euston maintains consistent superiority across all tested batch dimensions, proving the effectiveness of our

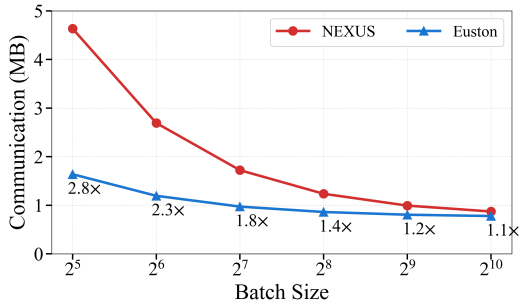
5. NEXUS identifies the price, which indicates the financial cost of outputting a word according to current AWS pricing policy.

TABLE 4: Inference accuracy of transformer models on the GLUE benchmarks.

Model	Dataset	Plaintext	NEXUS	Euston
BERT-Base	RTE	70.04%	69.88%	70.04%
	SST-2	93.23%	92.98%	93.12%
	QNLI	91.25%	90.85%	91.10%
LLAMA-3-8B	RTE	82.75%	81.24%	81.56%
	SST-2	94.94%	94.46%	94.67%
	QNLI	90.70%	90.20%	90.38%

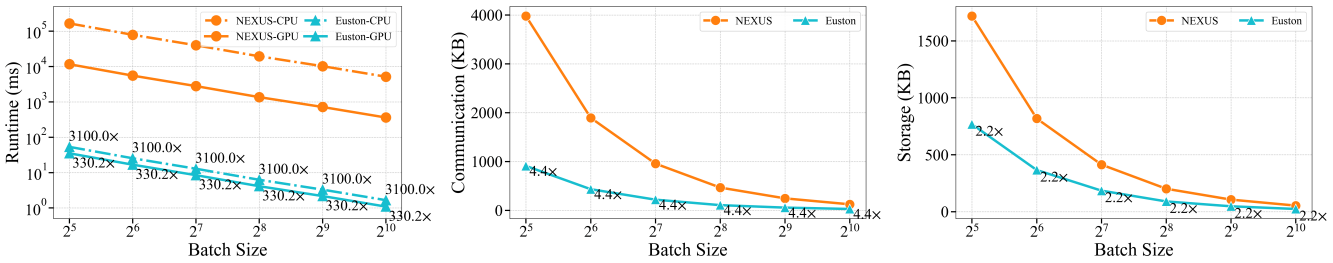


(a) Amortized runtime vs. batchsize.



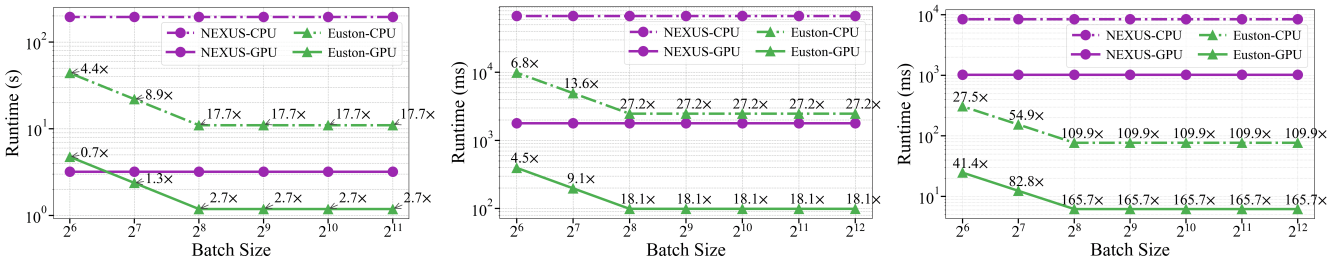
(b) Amortized communication size vs. batchsize.

Figure 3: Amortized overheads of Input-Weight Matrix Multiplication for different batchsizes in different settings. The dimensions of evaluated matrices are $\mathbb{R}^{128\times768}$ and $\mathbb{R}^{768\times768}$ (Similar to BERT-Base model).



(a) User's amortized runtime. (b) User's amortized communication size. (c) User's amortized storage size.

Figure 4: Amortized offline overheads of a user for different batchsizes in different settings. The dimensions of evaluated matrices are $\mathbb{R}^{128\times768}$ and $\mathbb{R}^{768\times768}$ (Similar to BERT-Base model).



(a) GELU amortized runtime. (b) LayerNorm amortized runtime. (c) Softmax amortized runtime.

Figure 5: Amortized runtime of nonlinear functions for different batch sizes in CPU and GPU environments. We set the token length to 128 and test using the computational scales of nonlinear functions in BERT-Base (GELU: 3072, LayerNorm: 768, Softmax: 128).

SVD-based ciphertext compression.

6.4.2. User Overhead. We focus on user-side offline overhead optimizations since online costs remain constant across batch sizes, which futher confirms the optimization bottleneck lies in the online phase. Fig. 4 quantifies Euston's user-side improvements over NEXUS across three dimensions:

Computational Overhead. Euston's user overhead stems primarily from SVD and SecComp of random matrices, while NEXUS requires costly SecDecomp operations. As shown in Fig. 4a, this achieves 3100× CPU acceleration and 330× GPU speedup, effectively democratizing access to resource-constrained devices. Crucially, user computation remains batchsize-agnostic as it depends solely on predefined matrix dimensions.

Communication Overhead. Our methods reduce transmission costs by 4.4× through selective encryption of singular values (Euston: $[\![\mathbf{D}]\!] + \mathbf{U} + \mathbf{V}$ vs. NEXUS: $[\![\mathbf{W}]\!] + [\![\mathbf{RW}]\!]$). This optimization is particularly impactful for edge devices with bandwidth constraints, as verified in Fig. 4b.

Storage Footprint. The user in Euston only maintains a random matrix $\mathbf{R}$, whereas NEXUS requires persistent storage of encrypted model parameters $[\![\mathbf{W}]\!]$. As shown in Fig. 4c, the storage footprint decreases 2.2× by eliminating user-side ciphertext weight caching.

These collective improvements enable practical deployment on resource-limited devices, with high-efficiency computation, communication, and storage.

6.4.3. Nonlinear Functions. Fig. 5 compares the amortized runtime of nonlinear operations between Euston and NEXUS under varying batch sizes. Our framework achieves asymptotic efficiency bounds at batch size 256 when fully utilizing the FHE ciphertext capacity (polynomial degree 2^{16} with 2^{15} slots), demonstrating two key operational regimes in different batch sizes t :

Underutilized Capacity ($t < 256$) Suboptimal packing efficiency occurs due to partial slot occupancy, particularly impacting GELU operations where Euston's GPU implementation shows 30% slower execution than NEXUS at $t = 64$.

Optimal Utilization ($t \geq 256$) Full ciphertext slot utilization maximizes throughput, demonstrating hardware acceleration ceilings: CPU optimizations reach 17.7× (GELU), 27.7× (LayerNorm), 109.9× (Softmax), while GPU implementations achieve 2.7× (GELU), 18.1× (LayerNorm), 165.7× (Softmax).

The superior performance derives from three architectural innovations: rotation-free column-packed ciphertext operations, depth-optimized approximations, and batched matrix processing. These results establish new state-of-the-art benchmarks for non-interactive nonlinear functions.

6.5. End-to-End Performance Evaluation

Consistent with established evaluation methodologies [9], [10], we analyze runtime breakdowns across secure

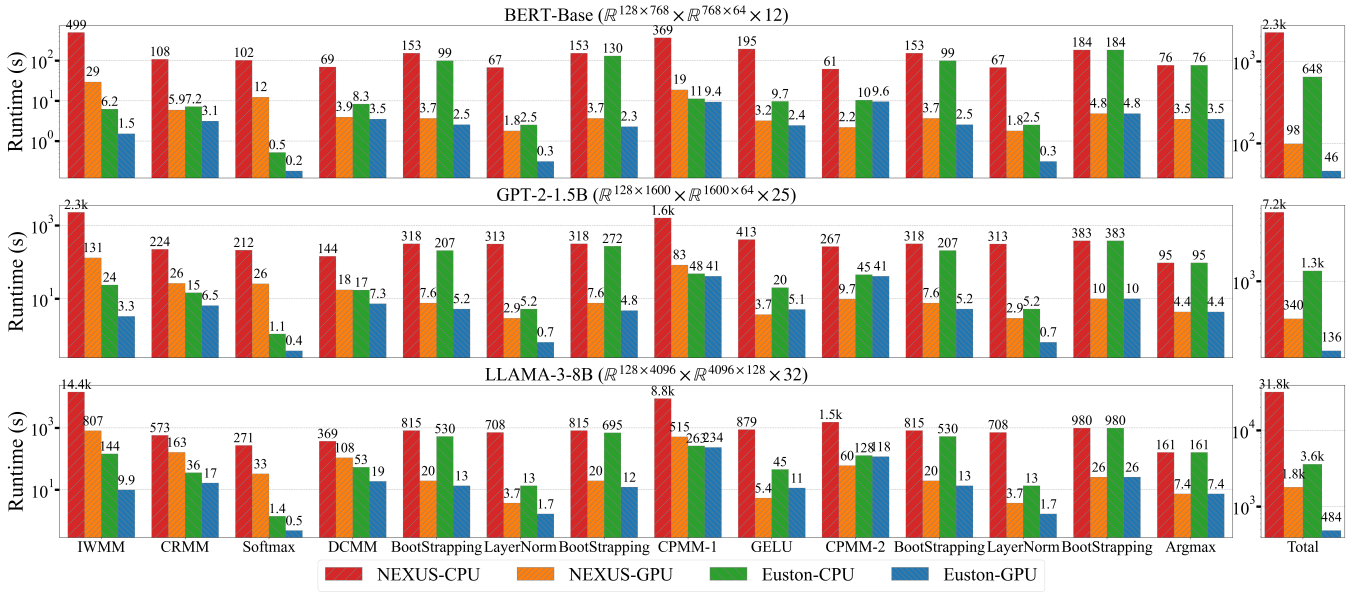


Figure 6: Comparison of runtime breakdown between Euston and NEXUS in CPU and GPU environment for BERT-Base, GPT-2-1.5B, and LLAMA-3-8B models. Note that we evaluate the amortized inference performance by batching 32 inputs consist of 128 tokens, which is different from the experiments in NEXUS—their LLAMA model only consist of 8 tokens. The detailed computational scale and depth consumption can be found in Appendix D.

inference phases. Fig. 6 compares Euston and NEXUS on BERT-Base, GPT-2-1.5B, and LLAMA-3-8B models under CPU/GPU configurations, with detailed computational scale and depth consumption analysis in Appendix D. At the system level, Euston demonstrates significant inference performance improvements over NEXUS across multiple language models, achieving a 3.5× speedup for BERT, 5.5× for GPT, and 8.8× for LLAMA on CPU platforms. On GPU implementations, Euston maintains robust optimizations with 2.1×, 2.5×, and 3.7× speedups for BERT, GPT, and LLAMA models, respectively.

Architectural Efficiency Drivers. The above benefits stem from dual optimizations in linear and nonlinear evaluations:

- **Linear Operation Enhancement**: Euston's user-oriented encryption strategy decouples ciphertext size from model parameters, combined with SVD-based transmission protocol. This achieves 80-100× CPU and 19.5-81.5× GPU acceleration in IWMM phase versus NEXUS' model-weight focused approach. [6]
- **Nonlinear Evaluation Optimization**: Column-packed ciphertext formats eliminates rotational operations in GELU, LayerNorm and Softmax evaluations compared to NEXUS' row-wise computation paradigm. The efficiency gap widens with model size, for example, exhibiting 26.9× (BERT) to 52.8× (LLAMA) reductions in LayerNorm overhead.

6. We recognize that there are some potential biases in HSMM processes (e.g., CRMM, DCMM, CPMM) stemming from NEXUS's unavailability of their source code related to these parts. Consequently, their results were estimated solely from published papers, whereas ours were empirically derived through actual experiments.

Depth Regulation Impact. Our adaptive depth regulation reduces bootstrapping overhead, achieving 1.25× (CPU) and 1.3× (GPU) acceleration in bootstrapping phases through depth-optimized HNEs.

Methodological Considerations. Minor performance discrepancies in CPMM-2 phases stem from incomplete NEXUS implementation details—our comparison uses their author-verified estimation methods. These variances constitute less total runtime and do not affect overall conclusions.

7. Related Work

7.1. Secure Neural Network Inference

Early research in secure neural network inference [7], [11], [33], [34], [35], [36], [37], [38] pioneered privacy-preserving techniques using technologies like FHE and MPC. A key foundational effort, CryptoNets [33], was the first to enable secure inference on encrypted data by adapting network structures to use square activation functions. Following this, GAZELLE [7] introduced a low-latency framework that innovatively combined FHE and two-party computation, designing efficient protocols to seamlessly switch between them. Addressing the high computational cost of non-linear functions, CrypTFlow2 [34] proposed novel two-party protocols that enabled the first secure inference on large-scale deep neural networks like ImageNet. To tackle the high communication overhead of mathematical functions in recurrent neural networks, SIRNN [36] introduced new protocols using lookup tables and mixed bit-width techniques. Building on these advancements, Cheetah

[11] further optimized efficiency for large models by designing FHE-based protocols that avoided costly rotations for linear layers. Representing a more recent trend, AutoFHE [38] focuses on automatically adapting convolutional neural networks for efficient evaluation under FHE. While these representative works have laid a solid foundation for STFI, significant challenges remain in efficiently handling complex operators like Softmax and LayerNorm.

7.2. Interactive Secure Transformer Inference

With the widespread adoption of Transformer models, significant research [8], [9], [18], [19], [20], [39], [40], [41], [42] has focused on developing specialized interactive protocols to enable secure Transformer inference. The first to build a complete framework for this was Iron [8], which used a hybrid FHE and MPC approach with novel compact packing to optimize protocols for key components like the attention mechanism and complex non-linear functions. Following this, Sigma [18] introduced the first end-to-end system based on Function Secret Sharing (FSS), creating new protocols and leveraging GPU acceleration to enable efficient secure inference on billion-parameter models. BOLT [9] further advanced the field by optimizing communication with improved HE techniques and achieving greater computational efficiency through highly accurate polynomial approximations and a token-pruning technique. Lastly, SecureGPT [39] represents a distinct multi-party approach, building a comprehensive suite of foundational MPC protocols to enable privacy-preserving inference across the entire GPT model architecture. Taking performance even further, BumbleBee [20] presented a two-party system that proposed new optimized protocols for both linear and non-linear layers, achieving significant speed and communication improvements over previous methods.

7.3. Non-interactive Secure Transformer Inference

To eliminate latency from frequent user-server interactions, NEXUS [10] proposed the first non-interactive protocol: 1) Implementing SIMD ciphertext compression and decompression algorithms in RNS-CKKS schemes using polynomial slot multiplexing, significantly reducing communication overhead; 2) Leveraging HE's self-homomorphic properties to introduce SIMD slot folding techniques and design efficient secure nonlinear protocols; 3) First integrating GPU implementation to substantially enhance inference efficiency. Our work builds primarily on NEXUS to further optimize the non-interactive inference performance and reduce the overheads for users.

8. Conclusion

In this paper, we propose Euston, a user-friendly and non-interactive STFI protocol that significantly outperforms the state-of-the-art solution NEXUS in both computational and communication efficiency, especially in terms of user overhead. We address the limitations of NEXUS by introducing an SVD-based secure transmission protocol, a carefully designed packing form for matrix products, and efficient nonlinear approximate computation with depth regulations. Our experimental results on various transformer models and datasets show that Euston achieves significant performance gains (up to 3100× lower user-side preprocessing costs and 8.8× higher system-wide inference performance) while maintaining high accuracy.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant No.62472431) and the National Key R&D Program of China (Grant 2024YFB2906803). The authors declare no competing financial interests. We thank the anonymous reviewers for their helpful comments, and our shepherd for the valuable guidance during the revision process.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 5998–6008.

[2] R. Lehmkuhl, P. Mishra, A. Srinivasan, and R. A. Popa, "Muse: Secure inference resilient to malicious clients," in *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. USENIX Association, 2021, pp. 2201–2218.

[3] X. Zhang, C. Chen, Y. Xie, X. Chen, J. Zhang, and Y. Xiang, "A survey on privacy inference attacks and defenses in cloud-based deep neural network," *Comput. Stand. Interfaces*, vol. 83, p. 103672, 2023.

[4] OpenAI, "Chatgpt: Optimizing language models for dialogue," Blog post, 2022.

[5] M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee, "Scalable extraction of training data from (production) language models," 2023.

[6] X. Jiang, M. Kim, K. E. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. ACM, 2018, pp. 1209–1222.

[7] C. Juvekar, V. Vaikuntanathan, and A. P. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*. USENIX Association, 2018, pp. 1651–1669.

[8] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, and T. Zhang, "Iron: Private inference on transformers," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[9] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider, "BOLT: privacy-preserving, accurate and efficient inference for transformers," in *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024, pp. 4753–4771.

[10] J. Zhang, X. Yang, L. He, K. Chen, W. Lu, Y. Wang, X. Hou, J. Liu, K. Ren, and X. Yang, "Secure transformer inference made non-interactive," in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025.

[11] Z. Huang, W. Lu, C. Hong, and J. Ding, "Cheetah: Lean and fast secure two-party deep neural network inference," in *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*. USENIX Association, 2022, pp. 809–826.

[12] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptol. ePrint Arch.*, p. 144, 2012.

[13] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*. ACM, 2012, pp. 309–325.

[14] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "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*, vol. 10624. Springer, 2017, pp. 409–437.

[15] A. C. Yao, "Protocols for secure computations (extended abstract)," in *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*. IEEE Computer Society, 1982, pp. 160–164.

[16] M. O. Rabin, "How to exchange secrets with oblivious transfer," *IACR Cryptol. ePrint Arch.*, p. 187, 2005.

[17] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*. ACM, 1987, pp. 218–229.

[18] K. Gupta, N. Jawalkar, A. Mukherjee, N. Chandran, D. Gupta, A. Panwar, and R. Sharma, "SIGMA: secure GPT inference with function secret sharing," *Proc. Priv. Enhancing Technol.*, vol. 2024, no. 4, pp. 61–79, 2024.

[19] Z. Li, K. Yang, J. Tan, W. Lu, H. Wu, X. Wang, Y. Yu, D. Zhao, Y. Zheng, M. Guo, and J. Leng, "Nimbus: Secure and efficient two-party inference for transformers," in *Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[20] W. Lu, Z. Huang, Z. Gu, J. Li, J. Liu, C. Hong, K. Ren, T. Wei, and W. Chen, "BumbleBee: Secure two-party inference framework for large transformers," in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025.

[21] E. Lee, J. Lee, J. No, and Y. Kim, "Minimax approximation of sign function by composite polynomial for homomorphic comparison," *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 6, pp. 3711–3727, 2022.

[22] J. H. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity," in *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, vol. 12492. Springer, 2020, pp. 221–256.

[23] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, vol. 10820. Springer, 2018, pp. 360–384.

[24] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, vol. 6841. Springer, 2011, pp. 505–524.

[25] C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, no. 4, pp. 656–715, 1949.

[26] L. H. Loomis, "Haar measure in uniform structures," 1949.

[27] G. B. Folland, *A course in abstract harmonic analysis*. CRC press, 2016.

[28] S. Halevi and V. Shoup, "Faster homomorphic linear transformations in helib," in *Annual International Cryptology Conference*. Springer, 2018, pp. 93–120.

[29] B. Möller, "Binary exponentiation," in *Encyclopedia of Cryptography and Security, 2nd Ed*. Springer, 2011, pp. 84–86.

[30] W. Cho, G. Hanrot, T. Kim, M. Park, and D. Stehlé, "Fast and accurate homomorphic softmax evaluation," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*. ACM, 2024, pp. 4391–4404.

[31] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *CoRR*, vol. abs/1606.08415, 2023.

[32] H. Yang, S. Shen, W. Dai, L. Zhou, Z. Liu, and Y. Zhao, "Phantom: A cuda-accelerated word-wise homomorphic encryption library," *IEEE Trans. Dependable Secur. Comput.*, vol. 21, no. 5, pp. 4895–4906, 2024.

[33] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. E. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, vol. 48. JMLR.org, 2016, pp. 201–210.

[34] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow2: Practical 2-party secure inference," in *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*. ACM, 2020, pp. 325–342.

[35] N. Jovanovic, M. Fischer, S. Steffen, and M. Vechev, "Private and reliable neural network inference," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1663–1677.

[36] D. Rathee, M. Rathee, R. K. K. Goli, D. Gupta, R. Sharma, N. Chandran, and A. Rastogi, "SiRnn: A math library for secure RNN inference," in *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 2021, pp. 1003–1020.

[37] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*. USENIX Association, 2020, pp. 2505–2522.

[38] W. Ao and V. N. Boddeti, "Autofhe: Automated adaption of cnns for efficient evaluation over FHE," in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. USENIX Association, 2024.

[39] C. Zeng, D. He, Q. Feng, X. Yang, and Q. Luo, "SecureGPT: A framework for multi-party privacy-preserving transformer inference in GPT," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 9480–9493, 2024.

[40] S. Balla and F. Koushanfar, "Heliks: He linear algebra kernels for secure inference," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 2306–2320.

[41] Y. Dong, W.-j. Lu, Y. Zheng, H. Wu, D. Zhao, J. Tan, Z. Huang, C. Hong, T. Wei, and W. Chen, "Puma: Secure inference of llama-7b in five minutes," *arXiv preprint arXiv:2307.12533*, 2023.

[42] T. Chen, H. Bao, S. Huang, L. Dong, B. Jiao, D. Jiang, H. Zhou, J. Li, and F. Wei, "THE-X: privacy-preserving transformer inference with homomorphic encryption," in *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*. Association for Computational Linguistics, 2022, pp. 3510–3520.

[43] A. Edelman and N. R. Rao, "Random matrix theory," *Acta numerica*, vol. 14, pp. 233–297, 2005.

[44] J. Rennie, "Jacobian of the singular value decomposition with application to the trace norm distribution," *//url http://qwone.com/~jason/writing/svdJacobian.pdf*, vol. 60, 2006.

[45] A. Edelman, "Eigenvalues and condition numbers of random matrices," *SIAM journal on matrix analysis and applications*, vol. 9, no. 4, pp. 543–560, 1988.

Appendix A. Detailed Proof of Lemma

Lemma 1. *For a uniform random matrix $\mathbf{R} \in \mathbb{R}^{m\times n}$ with i.i.d. entries following haar measure, let $\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ be its Singular Value Decomposition (SVD). Then:*

1) *The orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ are statistically independent of the singular value matrix $\mathbf{D}$ and of each other.*
2) *The distributions of the left and right singular vector matrices $\mathbf{U}$ and $\mathbf{V}$ follow the Haar measure on their respective orthogonal groups.*

Proof. We prove both statements using fundamental results from random matrix theory [43].

Part 1: Statistical Independence The proof of statistical independence relies on analyzing the joint probability density function (PDF) of the SVD components. Let the random matrix $\mathbf{R}$ have a PDF $f_{\mathbf{R}}(\mathbf{R})$. Since $\mathbf{R}$ is a "uniform random matrix", its density $f_{\mathbf{R}}(\mathbf{R})$ is constant over its domain.

1) *Joint distribution factorization*: The joint PDF of the SVD components $(\mathbf{U}, \mathbf{D}, \mathbf{V})$ can be expressed in terms of the density of $\mathbf{R}$ and the Jacobian of this transformation. Specifically, if $f_{\mathbf{U},\mathbf{D},\mathbf{V}}(\mathbf{u}, \mathbf{d}, \mathbf{v})$ denotes the joint PDF of $(\mathbf{U}, \mathbf{D}, \mathbf{V})$ at values $(\mathbf{u}, \mathbf{d}, \mathbf{v})$, then:
$$f_{\mathbf{U},\mathbf{D},\mathbf{V}}(\mathbf{u}, \mathbf{d}, \mathbf{v}) = f_{\mathbf{R}}(\mathbf{u}\mathbf{d}\mathbf{v}^\top) \cdot |J(\mathbf{u}, \mathbf{d}, \mathbf{v})| \quad (49)$$
where $|J(\mathbf{u}, \mathbf{d}, \mathbf{v})|$ is the absolute value of the Jacobian determinant of the transformation from $(\mathbf{U}, \mathbf{D}, \mathbf{V})$ to $\mathbf{R}$.
2) *Jacobian structure*: According to the fundamental result in random matrix theory concerning the SVD [44], [45], the Jacobian determinant $|J(\mathbf{U}, \mathbf{D}, \mathbf{V})|$ factors into terms depending on the singular values and terms depending on the orthogonal matrices, i.e.,
$$|J(\mathbf{U}, \mathbf{D}, \mathbf{V})| = C \cdot \left(\prod_{i<j} |\sigma_i^2 - \sigma_j^2| \right) \cdot g(\mathbf{U}) \cdot h(\mathbf{V}) \quad (50)$$
where σ_i are the singular values (entries of $\mathbf{D}$), C is a constant, and $g(\mathbf{U})$ and $h(\mathbf{V})$ are functions that depend only on $\mathbf{U}$ and $\mathbf{V}$ respectively.
3) *Density separation*: Since $f_{\mathbf{R}}(\mathbf{R})$ is a constant (let's denote it K) for a uniform random matrix $\mathbf{R}$, substituting this and the Jacobian structure (Eq. 50) into the joint PDF equation (Eq. 49), we get:
$$\begin{aligned} f_{\mathbf{U},\mathbf{D},\mathbf{V}}(\mathbf{u}, \mathbf{d}, \mathbf{v}) &= K \cdot C \cdot \left(\prod_{i<j} |\sigma_i^2 - \sigma_j^2| \right) \cdot g(\mathbf{u}) \cdot h(\mathbf{v}) \\ &\propto \left(\prod_{i<j} |\sigma_i^2 - \sigma_j^2| \right) \cdot g(\mathbf{u}) \cdot h(\mathbf{v}) \end{aligned} \quad (51)$$
For a continuous random matrix $\mathbf{R}$, the singular values are distinct with probability one. Consequently, the Jacobian term $\prod_{i<j} |\sigma_i^2 - \sigma_j^2|$ is non-zero, which permits the multiplicative separation of the joint probability density function into a product of terms, where each term depends exclusively on $\mathbf{D}$, $\mathbf{U}$, or $\mathbf{V}$. This separation implies that $f_{\mathbf{U},\mathbf{D},\mathbf{V}}(\mathbf{u}, \mathbf{d}, \mathbf{v}) = f_{\mathbf{U}}(\mathbf{u}) \cdot f_{\mathbf{V}}(\mathbf{v}) \cdot f_{\mathbf{D}}(\mathbf{d})$ for appropriately normalized marginal densities.
4) *Derivation of Independence*: From the above product form, we can derive the conditional PDF of $(\mathbf{U}, \mathbf{V})$ given $\mathbf{D} = \mathbf{d}$:
$$f_{\mathbf{U},\mathbf{V}|\mathbf{D}}(\mathbf{u}, \mathbf{v} \mid \mathbf{d}) = \frac{f_{\mathbf{U},\mathbf{D},\mathbf{V}}(\mathbf{u}, \mathbf{d}, \mathbf{v})}{f_{\mathbf{D}}(\mathbf{d})} = f_{\mathbf{U}}(\mathbf{u}) \cdot f_{\mathbf{V}}(\mathbf{v}) \quad (52)$$
Expressing this in terms of probability measures, we have:
$$\mathbb{P}(\mathbf{U}, \mathbf{V} \mid \mathbf{D} = \mathbf{d}) = \mathbb{P}(\mathbf{U}) \cdot \mathbb{P}(\mathbf{V}), \quad (53)$$
This result shows the statistical independence of $(\mathbf{U}, \mathbf{V})$ and $\mathbf{D}$. Besides, the conditional probability density of $(\mathbf{U}, \mathbf{V})$ factoring into the product of the marginal densities of $\mathbf{U}$ and $\mathbf{V}$ also implies that $\mathbf{U}$ and $\mathbf{V}$ are statistically independent of each other.

Part 2: Haar Measure Distribution This proof hinges on the orthogonal invariance of the distribution of $\mathbf{R}$, a property established by Assumption 3. This means that for any fixed orthogonal matrices $\mathbf{Q}_1 \in O(m)$ and $\mathbf{Q}_2 \in O(n)$, the transformed matrix $\mathbf{R}' = \mathbf{Q}_1\mathbf{R}\mathbf{Q}_2^\top$ has the same distribution as $\mathbf{R}$. We express this formally as $\mathbf{R}' \overset{d}{=} \mathbf{R}$.

1) *SVD under transformation*: Let $\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ be the SVD of $\mathbf{R}$, we have:
$$\mathbf{R}' = \mathbf{Q}_1(\mathbf{U}\mathbf{D}\mathbf{V}^\top)\mathbf{Q}_2^\top = (\mathbf{Q}_1\mathbf{U})\mathbf{D}(\mathbf{Q}_2\mathbf{V})^\top \quad (54)$$
Since $\mathbf{Q}_1\mathbf{U}$ and $\mathbf{Q}_2\mathbf{V}$ as products of orthogonal matrices are also orthogonal, $(\mathbf{Q}_1\mathbf{U}, \mathbf{D}, \mathbf{Q}_2\mathbf{V})$ forms a valid Singular Value Decomposition of $\mathbf{R}'$.
2) *Distributional equivalence*: Given that $\mathbf{R}'$ and $\mathbf{R}$ share the same distribution ($\mathbf{R}'$ and $\mathbf{R}$), their SVD components must also be identically distributed. This is because the SVD is a deterministic function mapping a matrix to its unique singular values and corresponding orthogonal matrices (assuming distinct singular values). Applying this transformation to identically distributed random variables preserves their joint distribution, thus:
$$(\mathbf{Q}_1\mathbf{U}, \mathbf{D}, \mathbf{Q}_2\mathbf{V}) \overset{d}{=} (\mathbf{U}, \mathbf{D}, \mathbf{V}) \quad (55)$$

3) *Haar measure characterization*: Leveraging the distributional equivalence $(\mathbf{Q}_1\mathbf{U}, \mathbf{D}, \mathbf{Q}_2\mathbf{V}) \overset{d}{=} (\mathbf{U}, \mathbf{D}, \mathbf{V})$ and the statistical independence of $(\mathbf{U}, \mathbf{V})$ from $\mathbf{D}$ (as established in Part 1), we deduce the properties of $\mathbf{U}$ and $\mathbf{V}$. Specifically, considering the joint distribution of the SVD components, we have:

$$\mathbf{Q}_1\mathbf{U} \overset{d}{=} \mathbf{U} \quad \text{and} \quad \mathbf{Q}_2\mathbf{V} \overset{d}{=} \mathbf{V} \tag{56}$$

for any fixed orthogonal matrix $\mathbf{Q}_1, \mathbf{Q}_2$. These marginal equality indicates that $\mathbf{U}$ and $\mathbf{V}$ is distributed identically to its transformation by $\mathbf{Q}_1$ and $\mathbf{Q}_2$, respectively. This invariance is the defining property of the Haar measure on the orthogonal group. Therefore, both $\mathbf{U}$ and $\mathbf{V}$ follow the Haar measure.

Thus, we have established both the statistical independence of $(\mathbf{U}, \mathbf{V})$ from $\mathbf{D}$ and the Haar measure properties of $\mathbf{U}$ and $\mathbf{V}$. □

Appendix B. Detailed Proof of Theorem

Theorem 1 (MM-IND Security of IWMM Protocol)**.** *Let* Π_{IWMM} *be the IWMM protocol where the user sends* $\{\mathbf{U}, \mathbf{V}, [\![\mathbf{D}]\!], \mathbf{X} = \mathbf{A} - \mathbf{R}\}$ *to the server, with* $\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ *being a uniformly random matrix. Under the assumptions 1, 2, 3 and Lemma 1, the protocol* Π_{IWMM} *satisfies MM-IND security. Specifically, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that:*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A},\Pi_{\mathsf{IWMM}}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\Pi_{\mathsf{IWMM}}}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda). \tag{19}$$

Proof. We proceed using a sequence of hybrid games. Let $\mathcal{A}$ be a PPT adversary. We aim to show that G_0 is computationally indistinguishable from G_3, where G_3 represents an ideal execution from the adversary's perspective.

Game G_0**:** This is the real protocol execution. The user sends $\{\mathbf{U}, \mathbf{V}, [\![\mathbf{D}]\!], \mathbf{X}\}$ to the server, where $\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ and $\mathbf{R}$ is a uniformly random matrix.

Game G_1**:** In this game, the encrypted singular values $[\![\mathbf{D}]\!]$ are replaced with encryptions of random values $[\![\widetilde{\mathbf{D}}]\!]$ such that $||\mathbf{D}|| = ||\widetilde{\mathbf{D}}||$ is maintained. The user sends $\{\mathbf{U}, \mathbf{V}, [\![\widetilde{\mathbf{D}}]\!], \mathbf{X}\}$.

Game G_2**:** In this game, the SVD components $(\mathbf{U}, \mathbf{V})$ are replaced with uniformly random orthogonal matrices $(\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}})$. The user sends $\{\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}}, [\![\widetilde{\mathbf{D}}]\!], \mathbf{X} = \mathbf{A} - \mathbf{R}\}$, where $\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ (note: $\mathbf{R}$ here is still derived from the original $\mathbf{U}, \mathbf{D}, \mathbf{V}$ before the replacement in the sent message).

Game G_3**:** In this game, the input matrix $\mathbf{A}$ is replaced with a random matrix $\widetilde{\mathbf{A}}$ from the corresponding set $\mathcal{S} = \{\mathbf{U}\widetilde{\mathbf{D}}\mathbf{V} + \mathbf{X} \mid \widetilde{\mathbf{D}} = \mathrm{diag}(d_1, \ldots, d_m),\ d_i \overset{\text{i.i.d.}}{\sim} \mathcal{U}\}$. The user then sends $\{\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}}, [\![\widetilde{\mathbf{D}}]\!], \widetilde{\mathbf{X}} = \widetilde{\mathbf{A}} - \mathbf{R}\}$.

Analysis of Indistinguishability between Games:

$G_0 \overset{c}{\approx} G_1$: The indistinguishability between G_0 and G_1 follows directly from the FHE IND-CPA security (Assumption 1). If an adversary could distinguish between the encryption of $\mathbf{D}$ and the encryption of $\widetilde{\mathbf{D}}$ (given that $||\mathbf{D}|| = ||\widetilde{\mathbf{D}}||$ ensures a valid encryption for the context), this would break the IND-CPA security of FHE. Therefore, $\left|\Pr[\mathsf{Exp}_{\mathcal{A},G_0}(1^\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A},G_1}(1^\lambda) = 1]\right| \leq \mathsf{negl}_1(\lambda)$.

$G_1 \overset{c}{\approx} G_2$: In both G_1 and G_2, the matrix $\mathbf{R}$ is defined as $\mathbf{U}\mathbf{D}\mathbf{V}^T$ and is a uniformly random matrix following the Haar measure. By Lemma 1, its SVD components $(\mathbf{U}, \mathbf{V})$ are Haar-distributed and statistically independent from $\mathbf{D}$. Consequently, Assumption 3 states that these Haar-distributed components $(\mathbf{U}, \mathbf{V})$ are computationally indistinguishable from uniformly random orthogonal matrices $(\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}})$, i.e., $(\mathbf{U}, \mathbf{V}) \overset{c}{\approx} (\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}})$. The only difference in the messages sent by the user between G_1 and G_2 lies in the distributions of these orthogonal matrices: G_1 sends $(\mathbf{U}, \mathbf{V})$, while G_2 sends $(\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}})$. Therefore, any adversary capable of distinguishing G_1 from G_2 could inherently distinguish $(\mathbf{U}, \mathbf{V})$ from $(\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}})$, which directly contradicts Assumption 3. Thus, $\left|\Pr[\mathsf{Exp}\mathcal{A}, G_1(1^\lambda) = 1] - \Pr[\mathsf{Exp}\mathcal{A}, G_2(1^\lambda) = 1]\right| \leq \mathsf{negl}_2(\lambda)$.

$G_2 \overset{c}{\approx} G_3$: The only difference between the messages sent by the user in G_2 and G_3 is whether the last component is $\mathbf{A} - \mathbf{R}$ or a random matrix $\widetilde{\mathbf{X}} = \widetilde{\mathbf{A}} - \mathbf{R}$. Since $\mathbf{R}$ is a uniformly random matrix, $\mathbf{X}$ is computationally indistinguishable from a random matrix $\widetilde{\mathbf{X}}$ due to the information-theoretic security of the one-time pad (Assumption 2). Therefore, $\left|\Pr[\mathsf{Exp}_{\mathcal{A},G_2}(1^\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A},G_3}(1^\lambda) = 1]\right| \leq \mathsf{negl}_3(\lambda)$.

Conclusion: By the triangle inequality, we can combine the indistinguishability of these sequential games: $\left|\Pr[\mathsf{Exp}_{\mathcal{A},G_0}(1^\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A},G_3}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$, where $\mathsf{negl}(\lambda) = \mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda) + \mathsf{negl}_3(\lambda)$ is also a negligible function.

In Game G_3, the adversary observes $(\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}}, [\![\widetilde{\mathbf{D}}]\!], \widetilde{\mathbf{X}})$. Crucially, all these components are either uniformly random (orthogonal matrices) or indistinguishably encrypted random values (diagonal matrices and masks). As such, they are entirely independent of the user's private input $\mathbf{A}$, revealing no information about it. This perfectly mirrors the definition of the ideal world execution, $\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\Pi_{\mathsf{IWMM}}}(1^\lambda)$, where the adversary learns nothing beyond what is permitted by the security definition (in this case, for MM-IND, zero information about the input $\mathbf{A}$).

Therefore, G_3 serves as the ideal experiment from the adversary's perspective: $\Pr[\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\Pi_{\mathsf{IWMM}}}(1^\lambda) = 1] = \Pr[\mathsf{Exp}_{\mathcal{A},G_3}(1^\lambda) = 1]$. Thus, we conclude that: $\left|\Pr[\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A},\Pi_{\mathsf{IWMM}}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\Pi_{\mathsf{IWMM}}}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$. This completes the proof of MM-IND security for the IWMM protocol.

□

TABLE 5: The scale and depth change of evaluations in transformer inference.

Operation	BERT-base	GPT-2-1.5B	LLAMA-3-8B	Depth Change	
				NEXUS	Euston
IWMM	$(\mathbb{R}^{128\times768}\times\mathbb{R}^{768\times768})\times3$	$(\mathbb{R}^{128\times1600}\times\mathbb{R}^{1600\times1600})\times3$	$(\mathbb{R}^{128\times4096}\times\mathbb{R}^{4096\times4096})\times3$	-1	-3
CRMM	$(\mathbb{R}^{128\times64}\times\mathbb{R}^{64\times128})\times12$	$(\mathbb{R}^{128\times64}\times\mathbb{R}^{64\times128})\times25$	$(\mathbb{R}^{128\times128}\times\mathbb{R}^{128\times128})\times32$	-1	-2
Softmax	$\mathbb{R}^{128\times128}\times12$	$\mathbb{R}^{128\times128}\times25$	$\mathbb{R}^{128\times128}\times32$	-16	-10
DCMM	$(\mathbb{R}^{128\times128}\times\mathbb{R}^{128\times64})\times12$	$(\mathbb{R}^{128\times128}\times\mathbb{R}^{128\times64})\times25$	$(\mathbb{R}^{128\times128}\times\mathbb{R}^{128\times128})\times32$	-1	-2
BootStrapping	$\mathbb{R}^{128\times768}$	$\mathbb{R}^{128\times1600}$	$\mathbb{R}^{128\times4096}$	$+16$	$+9$
LayerNorm	$\mathbb{R}^{128\times768}$	$\mathbb{R}^{128\times1600}$	$\mathbb{R}^{128\times4096}$	-16	-9
BootStrapping	$\mathbb{R}^{128\times768}$	$\mathbb{R}^{128\times1600}$	$\mathbb{R}^{128\times4096}$	$+16$	$+14$
CPMM-1	$\mathbb{R}^{128\times768}\times\mathbb{R}^{768\times3072}$	$\mathbb{R}^{128\times1600}\times\mathbb{R}^{1600\times6400}$	$\mathbb{R}^{128\times4096}\times\mathbb{R}^{4096\times14336}$	-1	-1
GELU	$\mathbb{R}^{128\times3072}$	$\mathbb{R}^{128\times6400}$	$\mathbb{R}^{128\times14336}$	-14	-12
CPMM-2	$\mathbb{R}^{128\times3072}\times\mathbb{R}^{3072\times768}$	$\mathbb{R}^{128\times6400}\times\mathbb{R}^{6400\times1600}$	$\mathbb{R}^{128\times14336}\times\mathbb{R}^{14336\times4096}$	-1	-1
BootStrapping	$\mathbb{R}^{128\times768}$	$\mathbb{R}^{128\times1600}$	$\mathbb{R}^{128\times4096}$	$+16$	$+9$
LayerNorm	$\mathbb{R}^{128\times768}$	$\mathbb{R}^{128\times1600}$	$\mathbb{R}^{128\times4096}$	-16	-9
BootStrapping	$\mathbb{R}^{128\times768}$	$\mathbb{R}^{128\times1600}$	$\mathbb{R}^{128\times4096}$	$+21$	$+21$
Argmax	$\mathbb{R}^{30522}$	$\mathbb{R}^{50257}$	$\mathbb{R}^{128256}$	$*$	$*$

Appendix C. Column-Packed and Row-Packed Matrix Multiplication

This matrix multiplication computes $\mathbf{P} \coloneqq \mathbf{Q} \cdot \mathbf{K}^\top$, where $\mathbf{K}$ is a column-packed ciphertext matrix as defined in Section 4.2.2. The operation essentially multiplies column-packed and row-packed matrices.

Plaintext Foundation. The product's diagonal elements are obtained by multiplying corresponding columns of $\mathbf{Q}$ with rows of $\mathbf{K}^\top$. Non-diagonal elements require cyclic shifts of the row-packed matrix. Consider two 2×2 matrices, let $\mathbf{P} \coloneqq [\mathbf{p}_1\ \mathbf{p}_2]$:

$$\begin{aligned}\mathbf{p}_1 &= \begin{bmatrix}q_{11}\\q_{21}\end{bmatrix} \odot \begin{bmatrix}k_{11}\\k_{21}\end{bmatrix} + \begin{bmatrix}q_{12}\\q_{22}\end{bmatrix} \odot \begin{bmatrix}k_{21}\\k_{22}\end{bmatrix}\\ \mathbf{p}_2 &= \begin{bmatrix}q_{11}\\q_{21}\end{bmatrix} \odot \begin{bmatrix}k_{21}\\k_{11}\end{bmatrix} + \begin{bmatrix}q_{12}\\q_{22}\end{bmatrix} \odot \begin{bmatrix}k_{22}\\k_{21}\end{bmatrix}\end{aligned} \tag{57}$$

Ciphertext Multiplication. Let the left-hand matrix denote as $[\![\mathbf{Q}]\!] = [[\![\mathbf{q}_1]\!], [\![\mathbf{q}_2]\!], \ldots, [\![\mathbf{q}_b]\!]]$ and the right-hand matrix $[\![\mathbf{K}^T]\!] = \begin{bmatrix}[\![\mathbf{k}_1]\!]\\[\![\mathbf{k}_2]\!]\\\vdots\\[\![\mathbf{k}_b]\!]\end{bmatrix}$, we learn from BOLT that it is feasible to compute diagonal-packed ciphertext matrices $[\![\mathbf{P}]\!] = \left[[\![\grave{\mathbf{p}}_1]\!], [\![\grave{\mathbf{p}}_2]\!], \ldots, [\![\grave{\mathbf{p}}_m]\!]\right]$, where

$$[\![\grave{\mathbf{p}}_i]\!] = \sum_{j=1}^{b} ([\![\mathbf{q}_j]\!] \boxtimes [\![\mathbf{k}_j]\!]^{\leftarrow(i-1)}), \quad i \in \{\widehat{m}\} \tag{58}$$

Batched Ciphertext Multiplication. This procedure computes the batched ciphertext multiplication like in Section 4.2.3. Building upon Equation (58), we construct each diagonal vector of $[\![\widehat{\mathbf{P}}]\!]$ as follows:

$$[\![\widehat{\mathbf{p}}_i]\!] = \sum_{j=1}^{b} ([\![\widehat{\mathbf{q}}_j]\!] \boxtimes [\![\widehat{\mathbf{k}}_j]\!]^{\Leftarrow(i-1)}), \quad i \in \{\widehat{m}\} \tag{59}$$

Appendix D. Computational Scale and Homomorphic Depth Analysis for Secure Inference Processes

In evaluating Euston's inference performance, we selected three distinct Transformer models: BERT-Base, GPT-2-1.5B, and LLAMA-3-8B. Table 5 presents the computational scale and homomorphic depth analysis for each process of these models, with some matrix multiplication processes abbreviated.

- IWMM: input matrix and weight matrix multiplication.
- CRMM: column-packed matrix and row-packed matrix multiplication.
- DCMM: diagonal-packed matrix and column-packed matrix multiplication.
- CPMM: column-packed matrix and plain unpacked matrix multiplication.

Appendix E.
Meta-Review

The following meta-review was prepared by the program committee for the 2026 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

E.1. Summary

The paper presents an approach for secure transformer inference using FHE, improving upon the state of the art by reducing both the amount of client computation (and storage) necessary for pre-processing, and the online computation cost significantly. It presents a novel preprocessing technique for secure matrix multiplication utilizing SVD decomposition and masking, as well as extensions of homomorphic matrix multiplication techniques to new ciphertext packing schemes. Through these, the paper delivers significant improvements in the efficiency of secure transformer inference over state of the art solutions.

E.2. Scientific Contributions

- Creates a New Tool to Enable Future Science
- Provides a Valuable Step Forward in an Established Field

E.3. Reasons for Acceptance

1) The paper provides a valuable step forward in an established and currently highly relevant field. While the absolute overheads remain high, this work takes an important step forward and makes secure transformer inference practical for larger models / more tokens than previously feasible.
2) The paper introduces a new SVD-based preprocessing technique for secure matrix multiplication that is likely to have applications beyond the immediate scope of FHE-based secure transformer inference.