CRAfT: Constant-Round Non-Polynomial Approximation and P2P Network for Secure Transformer Inference

Jinghao Zhao
zhaojinghao@stu.hit.edu.cn
Harbin Institute of Technology
Harbin, China

Hongwei Yang
yanghongwei@hit.edu.cn
Harbin Institute of Technology
Harbin, China

Xiaoyu Song
xiaoyu-song@stu.hit.edu.cn
Harbin Institute of Technology
Harbin, China

Meng Hao
haomeng@hit.edu.cn
Harbin Institute of Technology
Harbin, China

Hui He
hehui@hit.edu.cn
Harbin Institute of Technology
Harbin, China

Weizhe Zhang*
wzzhang@hit.edu.cn
Harbin Institute of Technology
Harbin, China

Abstract

Transformer models have recently emerged as a game-changing technology for applications in content generation. However, severe privacy concerns and the scalability bottlenecks of existing secure protocols limit their large-scale deployment. To address these challenges, we present CRAfT, an innovative system designed for high-scalability and low-latency secure Transformer inference. The core contributions of CRAfT are threefold: First, we propose a novel ciphertext packing strategy tailored for multiple mutually distrustful users. This approach completely eliminates the need for ciphertext rotations in linear projections layer and effectively offloads massive communication traffic to the high-speed P2P network. Second, we introduce a novel branch-free iterative fitting strategy based on trigonometric functions, which accurately approximates sigmoid-like functions (e.g., Tanh, Sigmoid, and erf in GELU). In the interval $[-3, 3]$, its approximation accuracy exceeds that of the plaintext Tanh-based GELU, while requiring only 4 communication rounds without relying on polynomial approximations. Finally, for complex operations like Softmax that exhibit partial sigmoid-like characteristics, we transform them into standard sigmoid-like functions, allowing us to directly apply our fitting strategy to minimize cryptographic costs while maintaining model accuracy. Comprehensive evaluations on BERT-base demonstrate that CRAfT increases multi-client inference throughput by up to 8.5× and reduces per-client WAN communication volume by 38.4% compared to the highly optimized baseline (BumbleBee).

Keywords

Secure Inference; Privacy-Preserving Machine Learning; Secure Multi-Party Computation; Homomorphic Encryption

1 Introduction

Transformer models, driven by the attention mechanism [2, 41], increasingly rely on cloud-based deployment paradigms for inference in crucial fields like natural language processing [1, 4, 7, 31, 40] and computer vision [5, 8, 24, 39]. However, the prevalent MLaaS paradigm raises severe privacy and security concerns [27, 37]: users are reluctant to upload private or sensitive data due to potential data breaches, and servers cannot deploy proprietary models to edge devices due to the risk of intellectual property theft. Therefore, it is imperative to adopt specialized methods to enhance the security of inference data, ensuring the safeguarding of both users' personal information and proprietary models.

Privacy-Preserving Machine Learning (PPML) provides a viable solution to this challenge. PPML aims to perform model evaluations while ensuring the user data and proprietary models remain private. With PPML, organizations can deploy and utilize Transformer models, facilitating various downstream tasks without privacy concerns.

Existing methods. Cryptographic protocols have demonstrated potential in secure inference. Existing methods advocate for performing secure inference using a standard single-client paradigm, a widely adopted architecture since it is easy to implement and allows the server to bear the primary bandwidth costs [6, 16, 19, 28]. To handle the complex non-linear functions in Transformer models, prior studies [25, 30, 32] proposed to first approximate these functions with piecewise polynomials, and then evaluate them securely using cryptographic protocols. Such methods achieved better results than previous approaches using simple low-order polynomials [11, 28].

Our Proposal. We highlight that this prevalent practice overlooks two critical opportunities for optimization: the abundant idle bandwidth resources among mutually untrusted clients within high-speed edge networks or co-located environments (where local P2P bandwidth significantly exceeds client-to-server WAN links), and trigonometric functions are inherently better suited than polynomials for approximating sigmoid-like functions. Building on this observation, we present CRAfT: a Constant-Round non-polynomial Approximation and P2P network for secure Transformer inference, which serves as an end-to-end solution that privately utilizes the P2P network, performs non-polynomial approximations, and significantly outperforms SOTA solutions.

CRAfT consists of three parts. First, we propose a data packaging method for multiple mutually untrusted clients within a P2P network, which eliminates ciphertext rotation in linear projections while leveraging P2P to offload a large amount of wide area network traffic to the P2P network. Secondly, CRAfT employs a branch-free iterative fitting strategy to extract the approximations of target sigmoid-like functions. By leveraging existing secure primitives for base trigonometric functions[45], this component iteratively approximates target functions, completely eliminating the ciphertext comparisons brought by traditional piecewise approximations.

*Corresponding author.

Finally, we transform the softmax into a sigmoid-like function and utilize our proposed fitting strategy for approximation, reducing computational costs while maintaining accuracy.

Compared to previous studies, CRAfT employs a more efficient architecture and approximation strategy for secure inference, which not only conserves communication resources and time but also achieves competitive inference performance in terms of both accuracy and latency.

Evaluations. We conduct comprehensive experiments to verify the effectiveness of our proposed system. By utilizing a novel P2P architecture, we can achieve superior scalability and throughput compared to existing solutions that rely on traditional client-server communication. Specifically, while single-client execution introduces a structural overhead, our method dramatically outperforms all baselines as the workload scales. On average, when serving 32 concurrent clients in end-to-end BERT-base inference under NET-A, the amortized execution time of our protocol is reduced to merely 90.3s per client, achieving an effective 6.9× throughput speedup over the SOTA method, i.e., BumbleBee. Besides, our P2P scheme achieves highly efficient bandwidth utilization, reducing the amortized WAN communication per client by 38.4% (dropping to just 3.88 GB) compared to the SOTA method.

Our experiments further explore the factors that make our nonlinear approximation protocols highly efficient and accurate. The results reveal that the success of our fitting strategy can be attributed to two aspects: (1) High precision over wide intervals. By optimizing the nesting depth (L), our method stably controls the maximum approximation error for activation functions like GeLU within the 10^{-4} range for standard intervals (e.g., $[-4.0, 4.0]$). Remarkably, even when scaling to an extraordinarily wide interval of $[-30.0, 30.0]$, increasing the depth to $L = 8$ allows the method to tightly bound the maximum errors to 1.65×10^{-5} for Tanh and 7.90×10^{-4} for GeLU. This robust global capability outperforms the ciphertext polynomial protocol (BumbleBee) by nearly two orders of magnitude. (2) Ultra-low communication overhead. Previous studies suffer from massive network round-trips in high-latency environments. In contrast, our protocol strictly bounds the communication round complexity to a constant 4 rounds. For Softmax evaluation, our advanced substitution function cuts WAN communication overhead by 49.8% and achieves a 10.86× execution speedup compared to BumbleBee, enhancing inference efficiency in bandwidth-constrained environments.

We also analyze the effects of our proposed non-linear replacements, specifically Softmax (S-Max) and Global LayerNorm (Global LN), on downstream task accuracy. We observe that the S-Max replacement serves as a universally applicable solution, closely aligning with the plaintext Teacher model across diverse datasets. For instance, on the IMDb and STS-B datasets, S-Max achieves competitive or nearly identical performance (92.15% and 89.03) compared to the baseline. However, the performance of Global LN exhibits higher task-specific variability. While it achieves marginal gains on specific tasks like IMDb (peaking at 92.26%), it experiences noticeable degradation on datasets like SST2 and MRPC.

In summary, our contributions are as follows:

- **Novel Ciphertext Packing and P2P Distribution.** We propose a collaborative architecture tailored for mutually distrustful clients. By introducing a novel ciphertext packing strategy, CRAfT completely eliminates the need for ciphertext rotations in linear projections. Furthermore, it offloads massive communication traffic to the high-speed P2P network, significantly alleviating the server's egress bandwidth bottleneck.
- **Branch-free Iterative Fitting via Trigonometric Functions.** We introduce a completely branch-free fitting strategy that requires exactly 4 communication rounds and eliminates the need for expensive ciphertext comparisons. Notably, within the $[-3, 3]$ interval, our method surpasses the precision of the standard plaintext Tanh-based GELU approximation. Furthermore, it gracefully extends to exceptionally wide intervals (e.g., $[-30, 30]$) while maintaining ultra-low approximation errors.
- **Crypto-Friendly Transformation for Complex Operations.** For computationally expensive operations like Softmax that exhibit partial sigmoid-like characteristics, we propose a novel transformation method. By converting them into standard sigmoid-like functions, we can directly apply our trigonometric fitting strategy. This approach maintains model accuracy while drastically reducing cryptographic overhead.
- **Comprehensive Experimental Validation.** Extensive evaluations demonstrate that our system significantly outperforms the SOTA baselines. Notably, it achieves a 6.9× throughput speedup and a 38.4% WAN bandwidth reduction for end-to-end BERT inference, while delivering orders-of-magnitude higher approximation precision and maintaining competitive downstream task accuracy.

2 Preliminaries

In this section, we introduce the notations, the threat model considered in our system, and the cryptographic primitives employed.

2.1 Notations

We denote by $\mathbb{N}$ the set of natural numbers. For an integer $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{0, 1, \ldots, n-1\}$. We use lowercase bold letters (e.g., $\mathbf{x}$) to represent column vectors and uppercase letters (e.g., X, W, Q, K, V) to represent matrices. The i-th element of a vector $\mathbf{x}$ is denoted by x_i. For a matrix W, its j-th column vector is denoted as $\mathbf{w}^{(j)}$, and the scalar element in its i-th row and j-th column is denoted as $w_{i,j}$ (or $W_{i,j}$). We denote the polynomial ring as $R = \mathbb{Z}[x]/(x^n + 1)$, where x is the polynomial variable and n (a power of 2) is the polynomial degree. For a modulus q, we denote $R_q = R/qR$ as the ring of polynomials with coefficients in $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$. For a polynomial $p(x) \in R_q$, we use $[p(x)]_j$ to denote the extraction of the coefficient of the x^j term. We use $\lfloor \cdot \rceil$, $\lfloor \cdot \rfloor$, and $\lceil \cdot \rceil$ to represent rounding to the nearest integer, floor, and ceiling operations, respectively. Since cryptographic primitives operate over integers, real-valued model parameters and intermediate data are mapped to integers using fixed-point arithmetic. A real value $r \in \mathbb{R}$ is encoded as an integer $\bar{r} = \lfloor r \cdot \Delta \rceil \bmod t$, where Δ is the scaling factor and t is the plaintext modulus. Note that multiplying two scaled numbers naturally results in a scale of Δ^2.

2.2 Threat Model

We consider a multi-user system consisting of a client set $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ and a centralized server $\mathcal{S}$. The clients possess private input data (e.g., sequence fragments), while the server facilitates the secure evaluation of the Transformer model. We adopt the semi-honest security model for all parties. Specifically:

- **Clients (including leechers and seeders):** Each client $u_i \in \mathcal{U}$ follows the protocol specification but attempts to infer information about other clients' data or the model parameters from the received messages. Notably, some clients voluntarily act as seeders. Unlike ordinary clients (leechers), seeders take on the additional responsibility of aggregating other clients' ciphertexts and forwarding the encrypted computation results from the server.
- **Server:** The server $\mathcal{S}$ faithfully executes the computation protocol but attempts to learn the clients' private inputs from the encrypted data or intermediate states.

Our security goal is to ensure that the server learns nothing about the clients' inputs other than the final inference result (if permitted), and that the clients learn nothing about each other's data or the server's private model weights beyond what is revealed by the final output.

2.3 Cryptographic Primitives

Our privacy-preserving system leverages lattice-based homomorphic encryption (HE) and secure multi-party computation techniques. We specifically differentiate between RLWE-based coefficient encoding, BFV-based slot encoding, and Additive Secret Sharing.

2.3.1 RLWE Encryption and Coefficient Encoding. Our security is based on the ring learning with errors problem [26]. To perform efficient linear algebra operations (e.g., matrix multiplication without rotation), we utilize coefficient encoding [16, 20, 25]. In this encoding, a column vector $\mathbf{v} \in \mathbb{Z}_q^n$ is mapped to a polynomial $v(x) = \sum_{i=0}^{n-1} v_i x^i \in R_q$. In our multi-user setting, a common random polynomial $a(x)$ is shared. Each user u_k holds a secret key $s_k(x)$ and an error polynomial $e_k(x)$. Throughout this paper, we use ct to denote a ciphertext. Subscripts are used to specify the encrypted content (e.g., $\mathrm{ct}_{Q,i}$ denotes a ciphertext encrypting the i-th column of matrix Q). We use $\odot$ to denote the homomorphic scalar multiplication between a ciphertext and a plaintext constant.

2.3.2 BFV Homomorphic Encryption and Slot Encoding. For element-wise arithmetic operations, we employ the Brakerski-Fan-Vercauteren (BFV) scheme [9]. BFV supports Single-Instruction-Multiple-Data (SIMD) operations via Slot Encoding. Using the Chinese Remainder Theorem, a plaintext vector $\mathbf{m} \in \mathbb{Z}_t^n$ is packed into n slots of a single polynomial. We denote the encryption function under slot encoding as $\mathsf{Enc}_{\mathrm{slot}}(\cdot)$. Operations on the resulting ciphertexts correspond to element-wise addition ($\oplus$) and element-wise multiplication ($\otimes$) on the underlying vector slots:

$$\mathrm{Dec}(\mathrm{ct}_a \oplus \mathrm{ct}_b) = \mathbf{a} + \mathbf{b} \bmod t, \tag{1}$$

$$\mathrm{Dec}(\mathrm{ct}_a \otimes \mathrm{ct}_b) = \mathbf{a} \otimes \mathbf{b} \bmod t. \tag{2}$$

This packing capability is essential for efficiently evaluating element-wise activation functions and attention score masking. For BFV ciphertexts, superscripts in parentheses indicate the key owner for encryption: $\mathrm{ct}^{(C)}$ and $\mathrm{ct}^{(S)}$ denote ciphertexts encrypted under the client's and server's secret keys, respectively. The encoding method (coefficient or slot) is clear from the context and the operations applied.

2.3.3 Additive Secret Sharing. To securely compute non-linear functions (e.g., trigonometric approximations) and avoid costly ciphertext-ciphertext multiplications, we combine HE with Additive Secret Sharing. A secret value x in a finite ring $\mathbb{Z}_t$ is additively split into two shares $\langle x \rangle_0$ and $\langle x \rangle_1$, held by the Client and Server respectively, such that $x = \langle x \rangle_0 + \langle x \rangle_1 \bmod t$. Neither party can infer the original value x from their individual share.

2.4 Truncation-Free Sine Function Evaluation Protocol

We utilize the sine function evaluation protocol [45] to securely compute $F(x) = a \cdot \sin(x)$. Let t be the plaintext modulus of the BFV scheme. Given an input $x \in \mathbb{Z}_t$ secret-shared with scale Δ^2, we have $x \equiv x_0 + x_1 \bmod t$. Over the integers, this implies $x_0 + x_1 = x + wt$ for some unknown wraparound integer $w \in \{0, 1\}$. The BFV-based protocol proceeds as follows:

Step 1 (Client): The Client computes its local trigonometric values with scale Δ, encrypts them under the BFV scheme, and sends $\mathrm{ct}_{s_0}, \mathrm{ct}_{c_0}$ to the Server:

$$s_0 \leftarrow \lfloor \Delta \sin(x_0/\Delta^2) \rceil \bmod t, \quad c_0 \leftarrow \lfloor \Delta \cos(x_0/\Delta^2) \rceil \bmod t. \tag{3}$$

Step 2 (Server): The Server computes its local values in $\mathbb{Z}_t$, absorbing the coefficient a:

$$s_1 \leftarrow \lfloor a\Delta \sin(x_1/\Delta^2) \rceil \bmod t, \quad c_1 \leftarrow \lfloor a\Delta \cos(x_1/\Delta^2) \rceil \bmod t. \tag{4}$$

It homomorphically evaluates the angle sum identity. The scale naturally multiplies to $\Delta \times \Delta = \Delta^2$:

$$\mathrm{ct}_y \leftarrow (\mathrm{ct}_{s_0} \otimes c_1) \oplus (\mathrm{ct}_{c_0} \otimes s_1). \tag{5}$$

The Server samples a random share $y_1 \in \mathbb{Z}_t$, blinds the ciphertext via $\mathrm{ct}_{y_0} \leftarrow \mathrm{ct}_y \ominus y_1$, and returns ct_{y_0} to the Client.

Step 3 (Client): The Client decrypts to obtain its final share in $\mathbb{Z}_t$:

$$y_0 \leftarrow \mathrm{Dec}(\mathrm{ct}_{y_0}). \tag{6}$$

Crucially, by setting the parameters such that the ratio of the plaintext modulus to the scale satisfies $t/\Delta^2 \approx 2k\pi$ (where $k \in \mathbb{Z}^+$), the term involving the wraparound w becomes $\sin(\cdots + w \cdot \frac{t}{\Delta^2}) \approx \sin(\cdots + w \cdot 2k\pi)$. Because of the sine function's periodicity, this modular wraparound error is naturally absorbed, leaving only a negligible quantization error. This approach elegantly bypasses the need for expensive secure truncation protocols while maintaining the output as valid additive secret shares over $\mathbb{Z}_t$.

3 P2P-based Matrix Multiplication Protocol

3.1 Linear Projections

3.1.1 Single-client encoding. In Transformer models, the linear transformations across various layers, including the generation of Query (Q), Key (K), and Value (V) matrices in the self-attention mechanism, as well as the feed-forward networks and output projections, can be uniformly described as the multiplication of an input matrix X and a weight matrix W. Let the input matrix be

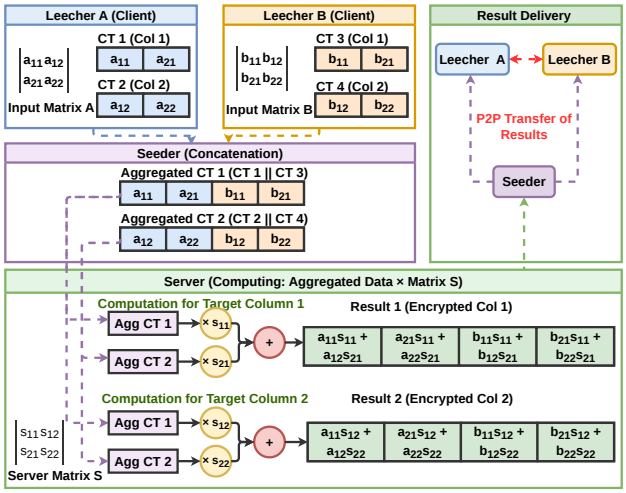

Figure 1: P2P-based Ciphertext Packaging Protocol.

$X \in \mathbb{Z}^{B\times d_{in}}$, where B denotes the total number of tokens and d_{in} denotes the input feature dimension; let the weight matrix be $W \in \mathbb{Z}^{d_{in}\times d_{out}}$. The computation goal is to obtain the matrix $Y = XW \in \mathbb{Z}^{B\times d_{out}}$. In the context of HE, assuming the polynomial ring is $R_q = \mathbb{Z}_q[x]/(x^n+1)$, we adopt a column vector encoding scheme to avoid complex ciphertext rotation operations. We treat the matrix X as a set of d_{in} column vectors, denoting the i-th column vector as $\mathbf{x}^{(i)} = [x_{0,i}, x_{1,i}, \ldots, x_{B-1,i}]^{\mathrm{T}}$. We independently encode each column vector $\mathbf{x}^{(i)}$ into a polynomial $m_i(x)$. Specifically, the element $x_{k,i}$ in the vector is mapped to the coefficient of the x^k term in the polynomial, as shown in the following equation:

$$m_i(x) = \sum_{k=0}^{B-1} x_{k,i} \cdot x^k. \tag{7}$$

The input matrix X is encoded into d_{in} ciphertexts $\{\mathrm{ct}_0, \ldots, \mathrm{ct}_{d_{in}-1}\}$. Based on the column space property of matrix multiplication, the j-th column of the output can be directly computed as a linear combination of the input ciphertexts and the plaintext weight scalars $w_{i,j}$:

$$\mathrm{ct}_{out,j} = \sum_{i=0}^{d_{in}-1} \left(\mathrm{ct}_i \odot w_{i,j}\right). \tag{8}$$

Since the HE scheme natively supports ciphertext-plaintext scalar multiplication ($\odot$) and ciphertext addition, the scalar $w_{i,j}$ is broadcast and multiplied by every coefficient of ct_i. This operation only scales the polynomial coefficients without altering the power of x, preserving the batch alignment of the data. Crucially, this method completely eliminates costly ciphertext rotations. Compared to slot encoding, it bypasses the most time-consuming HE operations (e.g., automorphisms and NTTs), significantly accelerating the linear layer inference.

3.1.2 Multi-user packaging. While computationally efficient, column vector encoding is highly sparse ($B \ll n$). To maximize ciphertext density and offload server connections, we adopt a high-speed P2P architecture, as illustrated in Figure 1: local nodes (Leechers) send valid data fragments to a Seeder, which packs them into a single dense ciphertext for the server.

Crucially, P2P networks inherently lack a trusted third party. Traditional ciphertext aggregation requires such a trust anchor to distribute a global public key for all users, making it fundamentally inapplicable here. To achieve secure aggregation in this decentralized environment, we propose a trust-anchor-free polynomial coefficient domain splicing mechanism based on a dual common reference. This scheme allows different users to encrypt with their own private keys, enabling the Seeder to aggregate all ciphertexts without decryption.

The system runs on the quotient ring $R_q = \mathbb{Z}_q[x]/(x^n+1)$. Following the standard common reference string model, we assume a set of independent, uniformly random polynomials $\{a_i(x)\}_{i=0}^{d_{in}-1}$ and a single polynomial $a'(x)$ are publicly available to all nodes, serving as the encryption bases for each feature dimension and the key-generation base, respectively. Assuming there are $|\mathcal{U}|$ users in the network, we partition the full set of n coefficient indices of the polynomial $\{0, 1, \ldots, n-1\}$ into $|\mathcal{U}|$ disjoint subsets $I_1, I_2, \ldots, I_{|\mathcal{U}|}$. Here, the set I_u represents the dedicated coefficient position indices assigned to the u-th user (corresponding to the offset of their data in the sequence of length B). Each user u generates a private key polynomial $s_u(x)$ and an error polynomial $e'_u(x)$, and publishes a public key derived from the key-generation base:

$$pk_u(x) = a'(x) \cdot s_u(x) + e'_u(x) \bmod q. \tag{9}$$

The user does not generate a complete RLWE ciphertext polynomial, but only computes specific coefficients belonging to their index set I_u. An RLWE ciphertext is a polynomial pair (c_0, c_1); in our scheme, the second component for dimension i is the global constant $c_{1,i}(x) = -a_i(x)$, shared by all users and requiring no per-user computation. For any index $k \in I_u$ and feature dimension i, the user calculates the first-component value $c^{(u)}_{0,k,i}$ at that position. This value is obtained by directly summing the polynomial product of the specific encryption base $a_i(x)$ under the private key, an independent error term $e_{u,i}(x)$, and the message $m_{u,k,i} \in \mathbb{Z}_q$ to be encrypted:

$$c^{(u)}_{0,k,i} = \left([a_i(x) \cdot s_u(x)]_k + [e_{u,i}(x)]_k + m_{u,k,i}\right) \bmod q, \tag{10}$$

where the symbol $[\cdot]_k$ denotes extracting the coefficient of the x^k term in the polynomial. Users only send the coefficient sets $\{c^{(u)}_{0,k,i} \mid k \in I_u\}$ to the aggregation node, thereby avoiding the redundant data overhead caused by transmitting the full polynomial. After receiving coefficient fragments from all users, the aggregation node performs a reconstruction operation in the coefficient domain. This operation does not involve addition or multiplication modulo q, but directly constructs a new aggregated polynomial $C^{agg}_{0,i}(x)$ for each dimension i:

$$[C^{agg}_{0,i}(x)]_k = c^{(u)}_{0,k,i}, \quad \text{for the unique } u \text{ satisfying } k \in I_u. \tag{11}$$

Once the aggregation node completes the aggregated ciphertexts ct^{agg}_i for all feature dimensions, the computation server can utilize the plaintext weight matrix W to perform the linear transformation. Since scalar multiplication scales both components uniformly, the

weighted sum yields:

$$C_{0,\text{out},j}(x) = \sum_{i=0}^{d_{in}-1} \left(C_{0,i}^{agg} \odot w_{i,j} \right), \; C_{1,\text{out},j}(x) = \sum_{i=0}^{d_{in}-1} \left(-a_i(x) \cdot w_{i,j} \right). \tag{12}$$

To prevent users from inferring the server's model weights W from the input-output pairs, the server must blind both ciphertext components of the computation result. Specifically, the server samples a uniformly random mask polynomial $R_j(x) \in R_q$ as its secret share, together with a short ephemeral polynomial $v_j(x)$ and short noise polynomials $e_{s0,u,j}(x)$, $e_{s1,j}(x)$. It constructs a mask ciphertext of R_j under each user's public key with the same v_j:

$$M_{0,u,j}(x) = pk_u(x) \cdot v_j(x) + e_{s0,u,j}(x) + R_j(x), \tag{13}$$

$$M_{1,j}(x) = -a'(x) \cdot v_j(x) + e_{s1,j}(x). \tag{14}$$

The server splices the user-specific $M_{0,u,j}$ in the coefficient domain to form $M_{0,\text{agg},j}(x)$:

$$[M_{0,\text{agg},j}(x)]_k = [M_{0,u,j}(x)]_k, \quad k \in I_u, \tag{15}$$

and subtracts the mask ciphertext from the homomorphic result component-wise:

$$\text{ct}'_{0,\text{out},j}(x) = C_{0,\text{out},j}(x) - M_{0,\text{agg},j}(x), \tag{16}$$

$$\text{ct}'_{1,\text{out},j}(x) = C_{1,\text{out},j}(x) - M_{1,j}(x). \tag{17}$$

Since the output ciphertext pair $(\text{ct}'_{0,\text{out},j}, \; \text{ct}'_{1,\text{out},j})$ returned by the server is identical for all users in the same network, the system naturally leverages P2P file-sharing mechanisms for the downstream phase. Instead of the server transmitting the large ciphertext individually to each user, the ciphertext is distributed via P2P (e.g., BitTorrent-like protocols), reducing the server's egress bandwidth requirements.

When user u needs to decrypt the calculation result, they perform standard RLWE decryption on the blinded ciphertext pair using their own private key $s_u(x)$:

$$D_u(x) = \text{ct}'_{0,\text{out},j}(x) + \text{ct}'_{1,\text{out},j}(x) \cdot s_u(x) \bmod q. \tag{18}$$

For the index position $k \in I_u$ exclusively assigned to user u, the encryption-base and key-generation-base components cancel out perfectly. This yields a secret share of the true output:

$$[D_u(x)]_k = m'_{u,k,j} - [R_j(x)]_k + \epsilon_k, \tag{19}$$

where $m'_{u,k,j} = \sum_i w_{i,j} \, m_{u,k,i}$ is the exact result of the linear transformation, and ϵ_k absorbs the accumulated small noise terms (which are eliminated by subsequent rounding). Thus, the protocol output successfully satisfies the secret-sharing property: user u's decrypted share, when combined with the server's mask $[R_j(x)]_k$, recovers the exact plaintext evaluation.

This P2P packaging architecture provides robust security guarantees across three dimensions. First, regarding input privacy, although users share the public encryption bases $\{a_i(x)\}$, each node secures its strictly assigned data slots using an independent secret key $s_u(x)$ and error term $e_{u,i}(x)$. This renders the aggregated ciphertexts computationally indistinguishable from uniform noise to both the Seeder and the Server. Second, the protocol prevents lateral eavesdropping among Leechers. If user u attempts to decrypt an index $l \in I_p$ belonging to another user p, the decryption yields:

$$[D_u(x)]_l = \left[\sum_{i=0}^{d_{in}-1} a_i(x) w_{i,j} s_p(x) + \dots \right]_l - \left[\sum_{i=0}^{d_{in}-1} a_i(x) w_{i,j} s_u(x) \right]_l. \tag{20}$$

Because $s_p(x)$ and $s_u(x)$ are independent random polynomials, and $s_p(x)$ is kept secret from user u, the bases fail to cancel out. Under the RLWE assumption, the recovered value at this position is computationally indistinguishable from pseudo-random noise. Finally, to protect the server's proprietary model, the mask polynomial $R_j(x)$ ensures that users only receive a masked secret share. This strictly prevents algebraic reverse-engineering of the model weights W from input-output pairs. A formal security proof is deferred to Appendix C.3.

3.2 Attention Scoring

To maximize encoding efficiency without increasing ciphertext count, we leverage slot encoding. We define H attention heads with query matrices $Q^{(h)} \in \mathbb{Z}^{B \times d_k}$. We utilize a column-based packing strategy: for a specific feature dimension i, we extract the i-th column from all H heads and concatenate them into a vector of length $H \times B$. This vector is packed into a single ciphertext $\text{ct}_{Q,i}$, organized in a block structure where slots 0 to $B-1$ contain Head 0, slots B to $2B-1$ contain Head 1, and so forth. Furthermore, if packing H heads still leaves available ciphertext slots empty, we densely pack multiple such cascaded vectors into the remaining gaps to fully saturate the ciphertext capacity. When utilizing this denser packing strategy, a homomorphic rotation and accumulation operation is subsequently applied to correctly aggregate the parallel partial sums.

The computation of $A = QK^T$ requires that for a target token position j and feature i, the query data for head h must be multiplied by the scalar key value $K_{j,i}^{(h)}$. We construct a specialized plaintext vector $\text{pt}_{K,j,i}$ mirroring the cascaded structure of $\text{ct}_{Q,i}$. The output ciphertext for position j is computed via:

$$\text{ct}_{\text{out},j} = \sum_{i=0}^{d_k-1} \left(\text{ct}_{Q,i} \otimes \text{pt}_{K,j,i} \right), \tag{21}$$

where $\otimes$ denotes element-wise multiplication. This ensures simultaneous, correct weighting for all heads within a single ciphertext.

Unlike the computation for linear projection layers, which prioritizes utilizing client bandwidth to alleviate server communication pressure, the computation of attention scores focuses on leveraging client computational resources to share the server's computational load. We transform complex ciphertext-ciphertext multiplications into multiple ciphertext-plaintext multiplications to achieve this, strategically offloading the primary computational burden to the client. To securely compute the full attention scores without relying on costly ciphertext-ciphertext multiplications, we utilize a hybrid protocol combining secret sharing and mask blinding.

In this protocol, $\text{ct}^{(C)}(\cdot)$ denotes encryption under the client's private key, while $\text{ct}^{(S)}(\cdot)$ denotes encryption under the server's private key. Initially, the server homomorphically subtracts its generated random masks M_Q and M_K from the pre-held ciphertexts $\text{ct}^{(C)}(Q)$ and $\text{ct}^{(C)}(K)$. To prevent any potential data leakage, all such random masks in this protocol are uniformly sampled from

the entire plaintext space. The server then sends the blinded ciphertexts $\mathrm{ct}^{(C)}(\hat{Q})$ and $\mathrm{ct}^{(C)}(\hat{K})$, along with the masks encrypted under its own key $\mathrm{ct}^{(S)}(M_Q)$ and $\mathrm{ct}^{(S)}(M_K)$, to the client. Based on this masked decomposition, the target matrix product expands into four terms:

$$QK^{\mathrm{T}} = (\hat{Q} + M_Q)(\hat{K} + M_K)^{\mathrm{T}} = \underbrace{\hat{Q}\hat{K}^{\mathrm{T}}}_{T_1} + \underbrace{\hat{Q}M_K^{\mathrm{T}}}_{T_2} + \underbrace{M_Q\hat{K}^{\mathrm{T}}}_{T_3} + \underbrace{M_Q M_K^{\mathrm{T}}}_{T_4}. \tag{22}$$

The client first decrypts $\mathrm{ct}^{(C)}(\hat{Q})$ and $\mathrm{ct}^{(C)}(\hat{K})$ to obtain the blinded plaintexts $\hat{Q}$ and $\hat{K}$. It then proactively handles the majority of the processing by computing the partial results for T_1, $\mathrm{ct}^{(S)}(T_2)$, and $\mathrm{ct}^{(S)}(T_3)$ using a dense slot-packing strategy. This process significantly relieves the server's computational pressure. The term T_4 comprises only server-side masks and is ignored by the client. To secure the result, the client generates a random plaintext matrix as its local additive secret share S_{client}, subtracts it from the aggregated partial results, and sends the resulting ciphertext $\mathrm{ct}_{\text{share}}$ back to the server.

Finally, the server reconstructs the complete encrypted result by adding the missing term T_4, which it computes locally in plaintext and encrypts. The final encrypted attention score is obtained by $\mathrm{ct}_{\text{final}} = \mathrm{ct}_{\text{share}} + \mathrm{ct}^{(S)}(T_4)$. At the protocol's conclusion, the system state satisfies the secret sharing property:

$$\mathrm{Dec}_S(\mathrm{ct}_{\text{final}}) + S_{\text{client}} = QK^{\mathrm{T}}. \tag{23}$$

Through this approach, the client's contribution to the intensive local computation ensures that neither party can independently recover the true attention scores or the weights, achieving secure computation with strictly bounded homomorphic depth and highly optimized ciphertext slot utilization.

4 Non-linear Function Evaluation Based on Trigonometric Iteration

Modern deep learning architectures extensively employ diverse sigmoid-like activation functions, including Tanh, GELU, and the error function. Existing privacy-preserving solutions primarily rely on piecewise polynomial approximations to handle these sigmoid-like functions. However, this indirect approach introduces significant computational and communication overhead. The bottleneck stems primarily from the secure comparison operations required to evaluate piecewise conditions. Since secure comparisons demand extensive multi-party online interactions, the communication rounds explode as the fitting depth increases.

To bypass the need for secure comparisons while accurately fitting standard sigmoid-like activations, we propose approximating them using a nested sine function:

$$F_L(x) = a_L \sin\Big(\dots a_1 \sin\big(a_0 x\big)\dots\Big). \tag{24}$$

Our approach leverages a highly efficient, truncation-free sine function evaluation protocol introduced in prior work [45] (detailed in Section 2.4). This cryptographic foundation computes the exact sine evaluation in a single communication round without requiring expensive truncation protocols. Furthermore, since this underlying protocol strictly preserves the input scale Δ^2 at its output, our nested sine evaluation can be seamlessly executed layer by layer without any additional scale-matching overhead.

A sigmoid-like function typically consists of saturated regions on both sides and a linear region in the center. We first analyze the saturated regions to understand how our proposed function achieves boundary fitting.

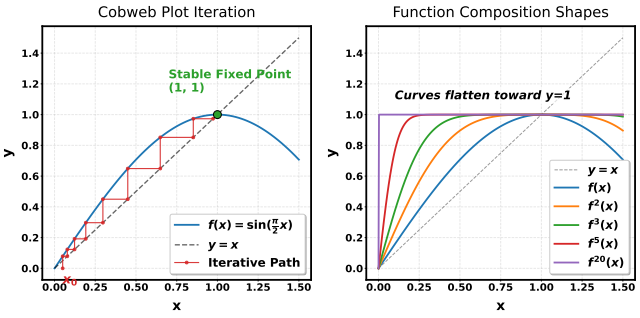

Figure 2: Analysis of the iterative system dynamics and fixed points (Left); Evolution of the function shape with increasing iteration depth (Right).

4.1 Saturated Regions Behavior

As illustrated in Figure 2 (Left), we first analyze the dynamical behavior of this iterative system. The figure displays the geometric relationship between the function $y = f(x) = \sin(\frac{\pi}{2}x)$ and the auxiliary line $y = x$. The system possesses two key fixed points within the interval $[0, 2]$: $x = 0$ and $x = 1$. At $x = 0$, the derivative of the function is $\frac{\pi}{2}$, which is greater than 1. According to fixed-point theory, this implies that $x = 0$ is an unstable repelling point. When the initial value x_0 acquires a tiny positive offset, the iteration process pushes the value away from the origin. Conversely, at $x = 1$, the derivative is 0. Since the absolute value of the derivative is less than 1, $x = 1$ constitutes a stable attractor. The red polyline in the figure records the iteration path starting from $x_0 = 0.05$; it can be seen that regardless of where the initial value lies within the $(0, 2)$ interval, the value eventually converges to 1 as the number of iterations increases.

Figure 2 (Right) further demonstrates the morphological changes of the composite function $y = f^{(n)}(x)$ as the number of iterations n increases. With deeper iteration, the function curve becomes increasingly flat near $x = 1$. This flatness stems from the zero derivative at $x = 1$, granting this point an extremely high convergence speed. Meanwhile, near $x = 0$, since the derivative is greater than 1, the curve slope grows exponentially with the number of iterations, thereby forming a steep rising edge.

This fitting method exhibits characteristics distinct from traditional methods when processing step-like functions. When using Fourier series to fit sigmoid-like functions, the Gibbs phenomenon [10] typically occurs near discontinuities. When using high-degree polynomial fitting, the Runge phenomenon [36] often appears at the interval edges, leading to significant numerical fluctuations. In contrast, the sine function iteration method proposed in this study maintains convergence within the interval $(0, 2)$. This means that the generated approximation function transitions smoothly during

the rise from 0 to 1 and does not produce overshoot or oscillation after reaching the steady-state value of 1. By controlling the number of iterations n, the steepness of the rising edge can be flexibly adjusted to adapt to the requirements of signal edge characteristics in different scenarios.

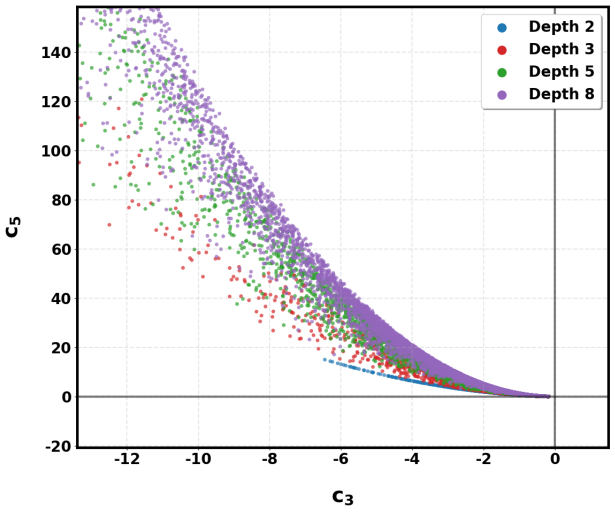

Figure 3: Taylor Coeffs (c_3, c_5) Comparison.

4.2 Linear Region Behavior

Next, we analyze the fitting of sigmoid-like functions near the origin using the proposed function. Common sigmoid-like functions include the hyperbolic tangent, Sigmoid, and the error function. Since the normalized target sigmoid-like functions and our proposed nested sine function are odd functions, their Taylor expansions lack even-order terms. Therefore, we analyze their morphology near the origin by focusing on the coefficients of the linear (c_1), cubic (c_3), and quintic (c_5) terms. Without loss of generality, to establish a unified mathematical framework for comparison, we assume that all target sigmoid-like functions discussed herein have been normalized. Specifically, they are scaled and shifted such that their output range is bounded within $[-1, 1]$, and their first derivative at the origin is exactly 1 (i.e., $c_1 = 1$). Correspondingly, to focus on the curvature characteristics of our proposed fitting function, we constrain its derivative at the origin to 1 as well, i.e., $F_L'(0) = \prod_{i=0}^{L} a_i = 1$.

Through numerical simulation, we calculated the distribution of coefficients under different parameter configurations. Figure 3 displays the scatter plot of the cubic term coefficients versus the quintic term coefficients of our proposed fitting function at different depths L. As shown, observing Figure 3 reveals that the scatter plot has a clear edge, indicating that the relationship between coefficients is constrained rather than arbitrarily distributed. In fact, strict mathematical constraints exist between the coefficients. To quantify this constraint, we define the coefficient ratio S as the ratio of the quintic term coefficient to the square of the cubic term coefficient, i.e., $S = \frac{c_5}{c_3^2}$.

Building upon the previous definitions, we formalize the boundary characteristics of the ratio S as follows:

Theorem 4.1 (Boundary Characteristics of Ratio S). *For any given depth $L \geq 1$ and parameter configuration $\{a_i\}$, the ratio S satisfies the following strict inequality:*

$$0.3 \leq S \leq 1.5 - \frac{1.2}{L}. \tag{25}$$

The lower bound equality $S = 0.3$ holds if and only if $L = 1$.

The detailed proof of this theorem is deferred to the Appendix. The inequality presented in Theorem 4.1 reveals a critical insight: as the function nesting depth L increases, the upper bound of the ratio S that our fitting function can achieve gradually expands and approaches 1.5. This asymptotic behavior is directly related to whether the function can accurately approximate specific sigmoid-like functions.

To demonstrate this, we can calculate the S values for common sigmoid-like functions via their Taylor expansions near the origin. For the normalized hyperbolic tangent function $\tanh(x)$, its expansion is given by:

$$\tanh(x) = x - \frac{1}{3}x^3 + \frac{2}{15}x^5 + O(x^7). \tag{26}$$

Here, the coefficients are $c_3 = -1/3$ and $c_5 = 2/15$, yielding a coefficient ratio $S = \frac{2/15}{(-1/3)^2} = 1.2$. Similarly, for the error function, after applying the aforementioned normalization to yield $f(x) = \mathrm{erf}(\frac{\sqrt{\pi}}{2}x)$, its expansion becomes:

$$\mathrm{erf}\left(\frac{\sqrt{\pi}}{2}x\right) = x - \frac{\pi}{12}x^3 + \frac{\pi^2}{160}x^5 + O(x^7). \tag{27}$$

This gives $c_3 = -\pi/12$ and $c_5 = \pi^2/160$, resulting in exactly $S = \frac{\pi^2/160}{(-\pi/12)^2} = 0.9$. Substituting these specific values into the boundary formula of the theorem reveals that when the depth L is small, the boundary upper limit is low and may not cover the S value of the target function. For instance, to reach 1.2 for $\tanh(x)$, according to the formula $1.5 - 1.2/L \geq 1.2$, we require L to reach at least 4. This signifies that as long as the depth of sine function iteration is moderately increased, typically to around 3 or 4 layers, the upper limit of the inequality can cover the characteristic values of common sigmoid-like functions. Consequently, with relatively few layers, our function can achieve an excellent fitting effect near the origin, serving as a basis for constructing high-precision approximation formulas.

4.3 Global Fitting Behavior

To transition from theoretical feasibility to practical instantiation, we formulate the search for the optimal parameter set $\{a_i\}$ as an optimization problem. For shallow layers and narrow intervals, we mainly employ a differential evolution algorithm to minimize the maximum absolute error between our nested sine function $F_L(x)$ and the target sigmoid-like function. For deeper layers and wider intervals, we utilize a Trust-Region-Reflective optimization based on numerical Jacobian estimation to efficiently navigate this highly non-linear problem. Taking the core component of GELU, $\mathrm{erf}(x/\sqrt{2})$, as a representative target, Figure 4 illustrates the optimized fitting curve and its corresponding error distribution. Crucially, the parameter optimization relies exclusively on data from

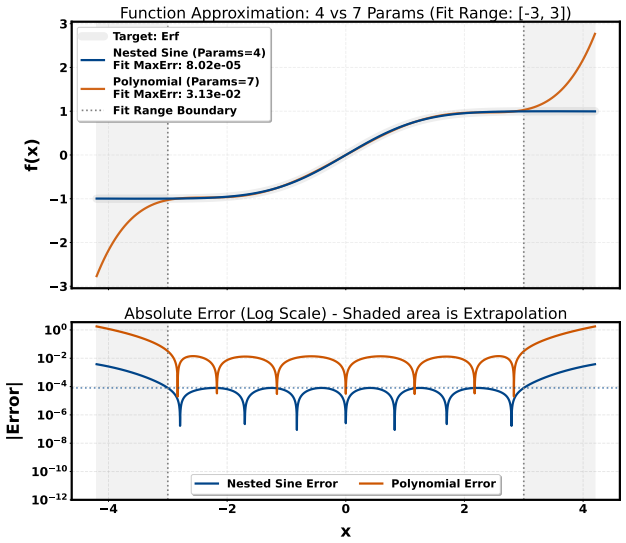

Figure 4: Fitting curve and error distribution for GELU approximation using differential evolution.

the central interval, making the outer regions a test of pure extrapolation. To objectively evaluate the performance of our proposed nested sine function, we introduce traditional polynomial fitting as a reference baseline in the experiment. To ensure the fairness of the comparison, the specific polynomial is configured to use a greater number of parameters than our nested sine function (even-degree terms of polynomials are useless for fitting). As shown in the figure, while high-degree polynomial fitting closely matches the target near the origin, it diverges significantly as the independent variable x extends outward, failing to capture the flat saturation regions. In contrast, our sine-based iteration maintains a high degree of coincidence across both the central linear region and the distant saturated regions.

Remarkably, in standard plaintext systems, the error function component of GELU is typically approximated using the composite function

$$\tanh\left[\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right]. \tag{28}$$

This heuristic approach is essentially an indirect approximation of the underlying erf function, which inherently bottlenecks its accuracy, resulting in a known maximum absolute error of 3.58×10^{-4}. However, rather than employing this indirect tanh-based approximation, our method directly approximates the exact $\mathrm{erf}(x/\sqrt{2})$ component using the optimized nested sine function. Evaluated over the core non-linear interval $[-3.0, 3.0]$, our proposed function at a mere nesting depth of $L = 3$ achieves a maximum absolute error of 1.62×10^{-4}.

5 Global Statistics-Based Optimization for Softmax and LayerNorm Layers

In the context of PPML, the Softmax layer in pre-trained models constitutes a significant bottleneck. The conventional Softmax function relies on a dynamic exponential sum denominator ($\sum e^{x_j}$), which necessitates complex multi-party communication, ciphertext synchronization, and highly expensive division operations.

To bypass this bottleneck, we propose a decoupled Softmax scheme based on global statistics. We introduce a running statistic Z for each attention head to track the expected value of the exponential sum. Once the fine-tuning completes, Z is fixed as a static constant, and the original Softmax is decoupled into an element-wise activation:

$$y' = \frac{e^{x_i}}{Z + e^{x_i}}, \tag{29}$$

which is called S-Max. This formulation, which resembles a shifted Sigmoid function, can be directly computed using the hyperbolic tangent (tanh) function via the equivalence $y' = \frac{1}{2}(1+\tanh(\frac{x-\ln Z}{2}))$. This transformation allows us to leverage the efficient tanh approximation method introduced earlier in this work, thereby fully eliminating intra-vector communication dependencies and reducing computational overhead.

To ensure smooth adaptation without disrupting pre-trained linguistic features, we employ a progressive distillation strategy. During the fine-tuning phase, the attention probability is computed as a convex combination of the standard Softmax and our decoupled formulation:

$$y_{train} = (1 - \alpha) \cdot \mathrm{Softmax}(x) + \alpha \cdot y', \tag{30}$$

where the transition weight α linearly scales from 0 to 1 over the initial training epochs.

Beyond the Softmax layer, we also explore simplifying the LN layer to mitigate the overhead of ciphertext division and square root operations. Following a similar rationale, we substitute the dynamic token-level mean and variance with global statistics (μ and σ^2) accumulated during fine-tuning. This allows the LN layer to be folded into a static linear transformation during inference:

$$y'_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma_i + \beta_i = W_{fixed} \cdot x_i + B_{fixed}, \tag{31}$$

which is called Global LN. The LN layer is similarly trained using the distillation strategy to ensure stability. While this linear approximation effectively eliminates the non-linear computation costs, empirical observations indicate a task-specific trade-off regarding model accuracy. Specifically, it maintains acceptable performance on tasks relying on macro-level semantics (e.g., IMDb, STS-B) but exhibits accuracy degradation on fine-grained syntactic tasks (e.g., MRPC). Therefore, rather than a universal replacement, this global LN variant serves as an optional acceleration strategy tailored for specific application scenarios where latency is the primary constraint.

6 Evaluation

6.1 Experimental Configurations

6.1.1 Experimental Environment and System Setup. The experiments in this study were deployed on a single server equipped with a 16-core CPU and 128GB of memory, with multi-threading optimization enabled. Unlike the single-client model commonly assumed in prior work, this experiment aims to evaluate the system's real-world performance under multi-client concurrent scenarios.

Table 1: Performance and Traffic Scalability Comparison under Two WAN Conditions.

Layer & Shape	System	Clients ($\|\mathcal{U}\|$)	WAN A (s)		WAN B (s)		Traffic (MB)	
			Total	Time/Client	Total	Time/Client	WAN	P2P
MLP Layer 1	BumbleBee	1	0.55	0.55	0.85	0.85	23.88	-
In: (128×768)	CRAfT	1	7.39	7.39	8.94	8.94	144.19	97
W: (768×768)		32	15.77	**0.49**	15.13	**0.47**	144.19	3,102
		60	20.00	**0.33**	21.49	**0.35**	144.19	5,816
MLP Layer 2	BumbleBee	1	1.26	1.26	1.94	1.94	32.61	-
In: (128×768)	CRAfT	1	22.66	22.66	27.33	27.33	432.47	385
W: (768×3072)		32	46.16	1.44	45.98	**1.44**	432.47	12,327
		60	59.92	**0.99**	64.14	**1.06**	432.47	23,113

In terms of network topology, this solution constructs two independent bandwidth-limited links through Traffic Control. First, all concurrent communications between clients (including the Seeder and multiple Leechers) and the server share a unified Wide Area Network (WAN) link. The experiment tests two configurations: WAN A (total bandwidth 1 Gbps, latency 50 ms) and WAN B (total bandwidth 500 Mbps, latency 100 ms). Second, P2P data distribution among clients utilizes a dedicated local network link (total bandwidth 10 Gbps, latency 1 ms). Regarding cryptographic configuration, BFV homomorphic encryption is employed with a polynomial degree $n = 8192$, plaintext/ciphertext moduli of 59 bits and 218 bits respectively, and 20-bit fixed-point precision is implemented. All initialization operations, such as transmitting keys and preparing masks, are not included in the overhead. We assume that all nodes in the P2P network are persistent and immune to failure or disconnection.

6.1.2 Fairness Constraints and System Evaluation. In real-world distributed deployments, the P2P architecture proposed in this study naturally aggregates massive edge computing power and independent physical network bandwidth as the number of participating nodes increases. However, to eliminate the hardware advantages associated with horizontal scaling, this experiment proactively forgoes the theoretical benefits of unlimitedly stacking computing and communication resources. This ensures a strictly fair comparison with baseline studies [25, 32] that are confined to single-machine architectures. Specifically, we deploy the server and all concurrent node processes on the same physical machine, forcing all entities to compete for the computing power of 16 CPU cores. Furthermore, regardless of the number of concurrent client nodes, the total external WAN traffic and internal P2P communication are strictly confined to predefined unified bandwidth caps. Crucially, the experiment employs only a single seeder, eliminating the possibility of alleviating bandwidth pressure through multiple data sources. This constraint mechanism completely neutralizes any performance gains derived from total resource expansion.

6.2 Microbenchmarks

6.2.1 Linear Operations. To evaluate the efficiency and scalability of our proposed P2P architecture, we analyze the execution time and network communication costs across two varying MLP layers under two different WAN conditions, as summarized in Table 1. When operating with a single client ($|\mathcal{U}| = 1$), the proposed method yields no advantage in communication volume and time because it only packs a single client's ciphertext.

However, the core advantage of our design emerges dramatically as the workload scales. By packing multiple clients' data concurrently (e.g., $|\mathcal{U}| = 32$ or 60), our method significantly minimizes the amortized execution time per client. For instance, in MLP Layer 1 under WAN B conditions, scaling to 60 clients drops the amortized time to 0.35s per client, delivering approximately a 2.4× speedup over BumbleBee's 0.85s. Similarly, in Layer 2 under WAN B, the amortized time decreases to 1.06s at $|\mathcal{U}| = 60$, outperforming the baseline's 1.94s (a 1.8× speedup). Consistent scalability trends are also observed under WAN A conditions, where the amortized time drops to 0.33s and 0.99s for Layer 1 and Layer 2, respectively.

This exceptional scalability is primarily driven by our network-offloading strategy. As observed in the traffic profile, the WAN communication footprint remains strictly constant, regardless of the increasing client count. Rather than congesting the bandwidth-constrained WAN, the linearly growing communication demands are entirely offloaded to the high-speed P2P network. Consequently, our P2P scheme effectively bypasses the wide-area bottleneck, ensuring highly parallelizable and efficient multi-client execution across varying network environments.

6.2.2 Non-linear Operations. Table 2 presents a comparison of the maximum approximation errors for the Tanh and GeLU functions across various intervals, evaluating the proposed method (at different nesting depths L) against existing ciphertext and plaintext baselines. Several key observations can be drawn from the experimental data:

First, the two activation functions exhibit different optimal responses to the approximation depth L. For the Tanh function, the approximation accuracy improves monotonically as L increases. When L increases from 2 to 4, the error drops by nearly two orders of magnitude (e.g., from 1.94×10^{-2} to 2.46×10^{-4} within the $[-3.0, 3.0]$ interval). Conversely, for the GeLU function, the approximation accuracy reaches its global optimum at $L = 3$ for standard intervals like $[-3.0, 3.0]$ and $[-4.0, 4.0]$ (achieving 1.20×10^{-4} and 3.03×10^{-4}, respectively). Interestingly, at these intervals, further increasing L to 4 slightly degrades performance due to the highly

Table 2: Maximum approximation errors of various methods for Tanh and GeLU activation functions across different evaluation intervals.

Func.	Interval	Method	Type	Error
Tanh	$[-3.0, 3.0]$	Nested Sine ($L = 2$)	C/P[a]	1.94e-2
		Nested Sine ($L = 3$)	C/P	6.19e-3
		Nested Sine ($L = 4$)	C/P	**2.46e-4**
		Rational Approx [3]	P	4.84e-2
	$[-4.0, 4.0]$	Nested Sine ($L = 2$)	C/P	2.66e-2
		Nested Sine ($L = 3$)	C/P	8.90e-3
		Nested Sine ($L = 4$)	C/P	**9.55e-4**
		Rational Approx [3]	P	1.18e-1
	$[-5.0, 5.0]$	Nested Sine ($L = 2$)	C/P	3.89e-2
		Nested Sine ($L = 3$)	C/P	1.10e-2
		Nested Sine ($L = 4$)	C/P	**1.94e-3**
		Rational Approx [3]	P	2.12e-1
	$[-30.0, 30.0]$	Nested Sine ($L = 8$)	C/P	**1.65e-5**
	Global	BOLT [32]	P[b]	6.60e-3
GeLU	$[-3.0, 3.0]$	Nested Sine ($L = 2$)	C/P	6.68e-3
		Nested Sine ($L = 3$)	C/P	**1.20e-4**
		Nested Sine ($L = 4$)	C/P	1.21e-3
	$[-4.0, 4.0]$	Nested Sine ($L = 2$)	C/P	1.85e-2
		Nested Sine ($L = 3$)	C/P	**3.03e-4**
		Nested Sine ($L = 4$)	C/P	1.44e-3
	$[-5.0, 5.0]$	Nested Sine ($L = 2$)	C/P	3.96e-2
		Nested Sine ($L = 3$)	C/P	6.64e-3
		Nested Sine ($L = 4$)	C/P	**1.41e-3**
	$[-30.0, 30.0]$	Nested Sine ($L = 8$)	C/P	**7.90e-4**
	Global	Tanh-Cubic [14]	P	**4.73e-4**
		Fast Sigmoid [14]	P	2.03e-2
		BOLT [32]	P[b]	9.36e-3
		BumbleBee [25]	P[b]	3.33e-2

[a] Type C/P indicates the error is identical for both Ciphertext and Plaintext. The quantization error introduced by converting floating-point to fixed-point in our ciphertext implementation is approximately 10^{-6}, which is negligible compared to the approximation error.

[b] The errors for BOLT and BumbleBee are evaluated based on their theoretical plaintext polynomial coefficients.

non-convex optimization landscape. However, for the extremely wide $[-5.0, 5.0]$ interval, the higher capacity of $L = 4$ becomes necessary, dropping the error to 1.41×10^{-3} compared to $L = 3$'s 6.64×10^{-3}. This suggests that $L = 3$ is optimal for standard GeLU distributions, while $L = 4$ provides necessary robustness for extreme outliers.

Second, the proposed method demonstrates remarkable robustness over wider approximation intervals, significantly outperforming traditional baselines. Existing rational approximation baselines (such as Rational Approx Deg 3/2, $(x + x^3/15)/(1 + 2x^2/5)$) degrade sharply from 4.84×10^{-2} to 2.12×10^{-1} when the interval is expanded from $[-3.0, 3.0]$ to $[-5.0, 5.0]$. Similarly, recent ciphertext polynomial protocols like BumbleBee exhibit a maximum error of 3.33×10^{-2} for GeLU. In contrast, by utilizing the optimal depth (i.e., $L = 4$ for Tanh and $L = 3$ or 4 for GeLU), the proposed method stably controls the maximum error within the 10^{-3} to 10^{-4} range even in the wide $[-5.0, 5.0]$ interval, outperforming BumbleBee by nearly two orders of magnitude. Furthermore, when extended to an extraordinarily wide interval of $[-30.0, 30.0]$, the proposed method scales gracefully by increasing the nesting depth to $L = 8$. Under this extreme setting, it tightly bounds the maximum errors to 1.65×10^{-5} for Tanh and 7.90×10^{-4} for GeLU, conclusively demonstrating its powerful capacity for high-precision global approximation.

Table 3: Micro-benchmark performance of non-linear layers in Transformer.

Op.	Method	Comm. (MB)	Rounds (Count)	WAN A Time(s)	WAN B Time(s)
GELU	BOLT	698.8	87	44.92	64.44
	BumbleBee	167.6	≈ 42[a]	4.99	10.51
	CRAfT	**158.4**	**4**	**3.70**	**5.40**
Softmax[b]	BOLT	642.56	232	32.15	62.75
	BumbleBee	157.88	≈ 205[a]	18.46	40.25
	CRAfT	**79.2**	**4**	**2.26**	**3.705**

[a] Not reported in [25]; estimated based on the latency difference between WAN A and WAN B.

[b] The Softmax evaluated in our method refers to the proposed S-max.

Finally, despite operating under ciphertext constraints, the proposed method achieves accuracy comparable to, or even exceeding, that of plaintext baselines. Taking the GeLU function as an example, widely used plaintext global approximation schemes (such as Tanh-Cubic, $0.5x(1 + \tanh(\sqrt{2/\pi}(x + 0.044715x^3)))$, and Fast Sigmoid, $x/(1 + \exp(-1.702x))$) yield maximum errors of 4.73×10^{-4} and 2.03×10^{-2}, respectively. Remarkably, the proposed method at $L = 3$ achieves a maximum error of 3.03×10^{-4} within the standard $[-4.0, 4.0]$ interval, which is strictly lower than the aforementioned plaintext baselines. This clearly demonstrates that the proposed method not only effectively overcomes the divergence issues typical of ciphertext polynomials over wide intervals but also possesses strong competitiveness in terms of absolute numerical accuracy.

To evaluate the efficiency of our proposed non-linear protocols, we compare our approach against state-of-the-art systems (BOLT and BumbleBee) under realistic WAN settings. As demonstrated in Table 3, our methods exhibit significant advantages. Crucially, our protocol strictly bounds the communication round complexity to an ultra-low 4 rounds. Since network round-trips heavily dominate the execution overhead in high-latency environments, this constant-round design acts as the primary driver behind our speedups. For GELU, our method achieves an execution time of 5.40 seconds in WAN B with a communication cost of 158.4 MB, resulting in a remarkable 1.95× and 11.93× speedup over BumbleBee and BOLT, respectively. Moreover, for Softmax evaluation, we employ an advanced alternative substitution function based on a shifted tanh evaluation. It should be noted that our performance gain stems fundamentally from protocol efficiency rather than mere algorithmic simplification: even if BOLT were adapted

to compute this exact same tanh function for a symmetric comparison, its underlying protocol would still require approximately 110 communication rounds, whereas our approach processes it strictly within 4 rounds. Consequently, our protocol drastically cuts the communication overhead down to 79.2 MB (saving approximately 49.8% compared to BumbleBee) and reduces the execution time to 3.705 seconds, achieving a 10.86× speedup over BumbleBee and a 16.93× speedup over BOLT. These results clearly demonstrate that our low-round, high-precision protocol is exceptionally robust and efficient for distributed inference in real-world high-latency network architectures.

6.3 Accuracy

In Table 4, we compare the task performance of our proposed Softmax (S-Max) and Global LayerNorm (Global LN) replacements against the baseline plaintext Teacher model. Following standard evaluation practices, we report the Pearson correlation for STS-B and accuracy for IMDb, SST2, and MRPC.

Overall, the proposed S-Max replacement demonstrates competitive performance, closely aligning with the Teacher model across most datasets. On IMDb, S-Max achieves an accuracy of 92.15%, slightly outperforming the Teacher's 91.92%. On STS-B, the performance is nearly identical to the baseline (89.03 vs. 89.05). While there is a slight performance drop on SST2 (0.57%) and a more noticeable degradation on MRPC (1.23%), S-Max generally maintains a robust approximation of the original model's capabilities.

In contrast, applying the Global LN replacement introduces higher task-specific variability. It performs exceptionally well on IMDb, achieving the highest accuracy of 92.26%, and maintains acceptable performance on STS-B (88.27). However, it suffers from more significant performance drops on SST2 (a 2.63% decrease) and MRPC (a 2.95% decrease) compared to the Teacher model. This indicates that the efficacy of Global LN is highly dataset-dependent.

Between the two methods, S-Max is a universally reliable replacement, offering greater stability and closer Teacher alignment across diverse tasks. Meanwhile, Global LN serves as an optional 'Aggressive Mode', maintaining comparable performance only on specific tasks.

Table 4: Task performance evaluation across datasets.

Dataset	Metric	Teacher	w/ S-Max	w/ Global LN
STS-B	Pearson ×100	89.05	89.03	88.27
IMDb	Accuracy (%)	91.92	92.15	92.26
SST2	Accuracy (%)	92.43	91.86	89.80
MRPC	Accuracy (%)	87.75	86.52	84.80

6.4 End-to-end Performance

To evaluate the real-world applicability and scalability of our proposed system, we conduct an end-to-end performance comparison on the BERT-base model (sequence length of 128) against several state-of-the-art baselines. We include THOR and SIGMA to provide a comprehensive evaluation across diverse cryptographic paradigms, highlighting the inherent trade-offs in secure inference. Specifically, FHE-based approaches like THOR excel in minimizing communication but generally require substantial computational resources, whereas FSS-based protocols like SIGMA offer highly efficient online phases but typically assume a trusted third party and involve large offline key material. It is important to note the different methodologies used to obtain the baseline data. Our protocol, BOLT, and BumbleBee are evaluated in the same experimental environment, with BumbleBee's word embedding overhead explicitly excluded for a fair comparison. In contrast, THOR's data is cited directly from its original paper, evaluated on an NVIDIA A100 GPU; because it only involves a single round of interaction, its communication overhead is considered negligible (≈ 0). SIGMA's communication overhead is taken from its original paper, while its execution times in WAN A and WAN B are theoretical values calculated based on the respective network bandwidths.

Table 5: End-to-end performance comparison on BERT-base (128 tokens).

System	Clients ($\|\mathcal{U}\|$)	Total Time (min)		WAN Comm. (GB)
		WAN A	WAN B	
THOR	1	10.04	-	≈ 0
SIGMA	1	≈ 4.50	≈ 9.27	34.67
BOLT	1	30.67	56.07	29.66
BumbleBee	1	10.50	17.10	6.30
Proposed	1	17.12	20.55	18.61
	8	23.27	30.52	42.43
	16	31.15	40.97	69.66
	32	48.18	64.12	124.11

Table 5 summarizes the total execution time (in minutes) across the two network conditions (WAN A and WAN B) and the total WAN communication overhead. When operating with a single client ($|\mathcal{U}| = 1$), our proposed protocol introduces a structural overhead compared to the highly optimized BumbleBee baseline. In WAN A, our method takes 17.12 minutes with 18.61 GB of WAN communication, compared to BumbleBee's 10.50 minutes and 6.30 GB. However, our protocol significantly outperforms BOLT, achieving nearly a 1.8× speedup in WAN A (17.12 min vs. 30.67 min) and a 2.7× speedup in WAN B (20.55 min vs. 56.07 min), while reducing the communication overhead by 37.3%. The initial overhead of our protocol at $|\mathcal{U}| = 1$ is an expected trade-off for establishing a highly scalable multi-client architecture.

The core advantage of our protocol emerges dramatically as the number of concurrent clients ($|\mathcal{U}|$) scales. While the total execution time naturally increases with more clients, the amortized time per client decreases significantly. For instance, when serving 32 clients concurrently in WAN A, the total execution time is 48.18 minutes. This translates to an amortized execution time of merely 1.51 minutes per client. Compared to BumbleBee's single-client execution time of 10.50 minutes, our scalable design delivers an effective 6.9× throughput speedup. A similar trend is observed in WAN B, where the amortized time drops to 2.00 minutes per client at $|\mathcal{U}| = 32$, achieving an 8.5× speedup over BumbleBee. Beyond execution time, our protocol demonstrates highly efficient bandwidth utilization. As the client count scales to $|\mathcal{U}| = 32$, the total WAN communication

reaches 124.11 GB. However, the amortized WAN communication per client drops to just 3.88 GB. This represents a 38.4% reduction in per-client WAN traffic compared to BumbleBee's 6.30 GB. This sub-linear scaling of WAN communication proves that our protocol effectively mitigates the network bottleneck, making it exceptionally well-suited for high-throughput, privacy-preserving inference in bandwidth-constrained environments.

7 Related Work

Due to the rapidly growing concerns about data privacy in machine learning, significant efforts have been made to design efficient cryptographic protocols for secure neural network inference. Early studies primarily focused on CNNs. Cryptonets [11] proposed one of the first protocols for HE-based private inference, followed by a long line of hybrid 2PC systems such as Gazelle [19], Delphi [28], and Cheetah [16], which combine HE for linear operations and Multi-Party Computation for non-linear activation functions. Compilers and optimized systems like Cryptflow [22], CrypTFlow2 [35], and SiRNN [34] further improved the 2PC evaluation of math functions. Recently, the remarkable success of Large Language Models has shifted the research focus towards Transformers. However, the secure evaluation of Transformers introduces unprecedented challenges due to their significantly larger parameter sizes and complex non-linearities.

Most existing studies on private Transformer inference operate in the two-party (2PC) semi-honest setting. Pure FHE-based systems, such as NEXUS [44], Powerformer [33], NeuJeans [18],and THOR [30], evaluate the entire Transformer using HE, but inherently suffer from high computational costs due to the high-order polynomial approximations required for non-linear layers. Hybrid 2PC systems offer a more practical trade-off. Early attempts like THE-X [6] bypass complex non-linearities by explicitly leaking intermediate plaintexts to the client, which completely breaks the security model. To compute these functions securely, other systems rely on polynomial approximations. However, MPCFormer [23] uses simple, non-piecewise polynomials, inevitably leading to noticeable accuracy degradation. To achieve faithful inference without fine-tuning, systems like Iron [13] utilize numerically precise protocols. BOLT [32] and BumbleBee [25] employ piecewise high-degree polynomial approximations. Notably, BumbleBee incurs higher fitting errors than BOLT to reduce communication overhead. In contrast, CipherGPT [15] adopts piecewise linear splines to approximate the curve within each interval. For linear layer evaluation, Iron's coefficient encoding suffers from sparse ciphertext utilization. To improve efficiency, BOLT achieves full slot utilization using SIMD packing with BSGS-optimized rotations, whereas Bumblebee mitigates this sparsity by compressing ciphertexts via its InterLeave algorithm. Furthermore, these 2PC systems are primarily designed for single-client interactions, which places a heavy communication burden on the server when scaled to multiple clients.

Research on multi-party privacy-preserving machine learning (> 2 parties, including the Trusted Third Party for FSS) follows two trajectories: Function Secret Sharing and Multi-Party Computation. In FSS-based optimization, Pika [42] minimizes communication via lookup tables for malicious environments. ORCA [17] integrates GPU acceleration to overcome storage bottlenecks, achieving sub-second image inference. SIGMA [12] supports secure inference for large-scale models like LLaMA2-13B through specialized Transformer protocols. Notably, these FSS schemes typically require a trusted third party for key distribution. Regarding broader MPC systems, ABY3 [29] established the foundation for three-party hybrid protocol switching. SecureNN [43] optimized neural network efficiency by eliminating garbled circuits, while CryptGPU [38] leveraged GPU kernels for ImageNet-scale training. Finally, SWIFT [21] provides enhanced robustness against malicious attacks.

8 Conclusion

In this paper, we presented CRAfT, an innovative system for high-scalability and low-latency secure Transformer inference, addressing the privacy concerns and scalability bottlenecks of existing protocols. By introducing a P2P data distribution architecture, CRAfT offloads public server data transmission to the client network, significantly alleviating server bandwidth constraints. Furthermore, we proposed a novel branch-free iterative fitting strategy based on trigonometric functions. This approach achieves high-precision approximation for sigmoid-like functions in constant communication rounds, avoiding expensive comparison protocols. Coupled with our MPC-friendly replacements for complex operations like Softmax, CRAfT significantly outperforms state-of-the-art systems in realistic network settings, achieving substantial reductions in both communication volume and end-to-end latency.

Despite these promising results, our system has certain limitations. First, CRAfT's efficiency decreases when the client count is small due to the overhead of computing on sparsely packed ciphertexts. In future work, we plan to design an adaptive mechanism that falls back to traditional single-client protocols in such scenarios. Second, to bypass the prohibitive communication overhead of multi-party dynamic statistics (e.g., in Softmax or Layer Normalization), CRAfT replaces them with static global approximations. We must acknowledge that relying on these global statistics introduces substantial limitations: it strictly necessitates a task-specific fine-tuning phase to calibrate these values, limiting the system's off-the-shelf applicability. Furthermore, this static approximation inevitably translates to a tangible accuracy degradation, particularly on fine-grained syntactic tasks. Accordingly, extending our cryptographic protocols to efficiently compute these dynamic statistics on the fly will be a major focus of our future work, aiming for zero-accuracy-loss secure inference across all downstream tasks.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] George A. Baker, Jr. and John L. Gammel. 1961. The Padé approximant. *J. Math. Anal. Appl.* 2, 1 (1961), 21–30.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Vol. 33. 1877–1901.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Springer, 213–229.

[6] Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. THE-X: Privacy-preserving transformer inference with homomorphic encryption. In *Findings Assoc. Comput. Linguist. (ACL)*. 3510–3520.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. Conf. North Am. Chapter Assoc. Comput. Linguist. (NAACL)*. 4171–4186.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[9] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. *Cryptol. ePrint Arch., Paper* (2012).

[10] J. Willard Gibbs. 1899. Fourier's series. *Nature* 59, 1539 (1899), 606–606.

[11] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proc. Int. Conf. Mach. Learn. (ICML)*. 201–210.

[12] Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. 2023. Sigma: Secure GPT inference with function secret sharing. *Cryptol. ePrint Arch., Paper* (2023).

[13] Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. 2022. Iron: Private inference on transformers. In *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Vol. 35. 15718–15731.

[14] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415* (2016).

[15] Xiaoyang Hou, Jian Liu, Jingyu Li, Yuhan Li, Wen-jie Lu, Cheng Hong, and Kui Ren. 2023. CipherGPT: Secure two-party GPT inference. *Cryptol. ePrint Arch., Paper* (2023).

[16] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. 2022. Cheetah: Lean and fast secure two-party deep neural network inference. In *Proc. 31st USENIX Secur. Symp. (USENIX Secur.)*. 809–826.

[17] Neha Jawalkar, Kanav Gupta, Arkaprava Basu, Nishanth Chandran, Divya Gupta, and Rahul Sharma. 2024. Orca: FSS-based secure training and inference with GPUs. In *Proc. IEEE Symp. Secur. Privacy (SP)*. IEEE, 597–616.

[18] Jae Hyung Ju, Jaiyoung Park, Jongmin Kim, Minsik Kang, Donghwan Kim, Jung Hee Cheon, and Jung Ho Ahn. 2024. Neujeans: Private neural network inference with joint optimization of convolution and FHE bootstrapping. In *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. 4361–4375.

[19] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. GAZELLE: A low latency framework for secure neural network inference. In *Proc. 27th USENIX Secur. Symp. (USENIX Secur.)*. 1651–1669.

[20] Dongwoo Kim and Cyril Guyot. 2023. Optimized privacy-preserving CNN inference with fully homomorphic encryption. *IEEE Trans. Inf. Forensics Security* 18 (2023), 2175–2187.

[21] Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. 2021. SWIFT: Super-fast and robust privacy-preserving machine learning. In *Proc. 30th USENIX Secur. Symp. (USENIX Secur.)*. 2651–2668.

[22] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2020. CryptFlow: Secure TensorFlow inference. In *Proc. IEEE Symp. Secur. Privacy (SP)*. IEEE, 336–353.

[23] Dacheng Li, Rulin Shao, Hongyi Wang, Han Guo, Eric P. Xing, and Hao Zhang. 2023. MPCFormer: Fast, performant and private transformer inference with MPC. In *Proc. Int. Conf. Learn. Represent. (ICLR)*.

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*. 10012–10022.

[25] Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. 2023. Bumblebee: Secure two-party inference framework for large transformers. *Cryptol. ePrint Arch., Paper* (2023).

[26] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On ideal lattices and learning with errors over rings. In *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Tech. (EUROCRYPT)*. Springer, 1–23.

[27] Fatemehsadat Mireshghallah, Mohammadkazem Taram, Praneeth Vepakomma, Abhishek Singh, Ramesh Raskar, and Hadi Esmaeilzadeh. 2020. Privacy in deep learning: A survey. *arXiv preprint arXiv:2004.12254* (2020).

[28] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A cryptographic inference system for neural networks. In *Proc. Workshop Privacy-Preserv. Mach. Learn. Pract. (PPMLP)*. 27–30.

[29] Payman Mohassel and Peter Rindal. 2018. ABY3: A mixed protocol framework for machine learning. In *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. 35–52.

[30] Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. 2025. THOR: Secure transformer inference with homomorphic encryption. In *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. 3765–3779.

[31] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Vol. 35. 27730–27744.

[32] Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. 2024. BOLT: Privacy-preserving, accurate and efficient inference for transformers. In *Proc. IEEE Symp. Secur. Privacy (SP)*. 4753–4771.

[33] Dongjin Park, Eunsang Lee, and Joon-Woo Lee. 2025. Powerformer: Efficient and high-accuracy privacy-preserving language model with homomorphic encryption. In *Proc. Annu. Meet. Assoc. Comput. Linguist. (ACL)*. 11090–11111.

[34] Deevashwer Rathee, Mayank Rathee, Rahul Kranti Kiran Goli, Divya Gupta, Rahul Sharma, Nishanth Chandran, and Aseem Rastogi. 2021. SIRNN: A math library for secure RNN inference. In *Proc. IEEE Symp. Secur. Privacy (SP)*. IEEE, 1003–1020.

[35] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2020. CrypTFlow2: Practical 2-party secure inference. In *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. 325–342.

[36] Carl Runge et al. 1901. Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten. *Zeitschrift für Mathematik und Physik* 46, 224-243 (1901), 20.

[37] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. 1310–1321.

[38] Sijun Tan, Brian Knott, Yuan Tian, and David J Wu. 2021. CryptGPU: Fast privacy-preserving machine learning on the GPU. In *Proc. IEEE Symp. Secur. Privacy (SP)*. IEEE, 1021–1038.

[39] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *Proc. Int. Conf. Mach. Learn. (ICML)*. PMLR, 10347–10357.

[40] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Vol. 30. 5998–6008.

[42] Sameer Wagh. 2022. Pika: Secure computation using function secret sharing over rings. In *Proc. Priv. Enhancing Technol. (PoPETs)*.

[43] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2019. SecureNN: 3-party secure computation for neural network training. In *Proc. Priv. Enhancing Technol. (PoPETs)*.

[44] Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen-jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. 2024. Secure transformer inference made non-interactive. *Cryptol. ePrint Arch., Paper* (2024).

[45] Jinghao Zhao, Hongwei Yang, Bobo Wang, Lichunxi Yang, Juncheng Li, Xiangrui Zeng, Meng Hao, Desheng Wang, Hui He, and Weizhe Zhang. 2026. PrivaLean: low-latency and high-accuracy system for secure 2PC inference. *Cryptol. ePrint Arch., Paper* (2026).

A Proof of the Coefficient Ratio Bound for Nested Sine Functions

Proof of Theorem 4.1. Let the fitting function $F_L(x)$ be composed of $L \geq 1$ nested sine functions, defined as:

$$F_L(x) = a_L \sin\Big(a_{L-1} \sin\big(\dots a_1 \sin(a_0 x) \dots\big)\Big),$$

where $\{a_i\}_{i=0}^{L}$ are non-zero real parameters. Assume the normalization condition at the origin:

$$F_L'(0) = \prod_{i=0}^{L} a_i = 1.$$

The Maclaurin series expansion of $F_L(x)$ is given by:

$$F_L(x) = x + c_3 x^3 + c_5 x^5 + O(x^7),$$

where the coefficient ratio is defined as $S = c_5/c_3^2$.

To determine the bounds of S, we define the squared cumulative gain sequence $w_k > 0$ for $k = 1, 2, \dots, L$:

$$w_k = (a_0 a_1 \dots a_{k-1})^2.$$

Using the Taylor expansion $\sin(u) \approx u - \frac{1}{6}u^3 + \frac{1}{120}u^5$, we substitute it into the nested structure layer by layer. Merging terms of the

same power yields the cubic and quintic coefficients:

$$c_3 = -\frac{1}{6}\sum_{k=1}^{L} w_k,$$

and

$$c_5 = \frac{1}{120}\sum_{k=1}^{L} w_k^2 + \frac{1}{12}\sum_{1\le j<k\le L} w_j w_k.$$

Using the symmetric polynomial identity

$$\sum_{j<k} w_j w_k = \frac{1}{2}\left[\left(\sum w_k\right)^2 - \sum w_k^2\right],$$

we rewrite c_5:

$$c_5 = \frac{1}{24}\left(\sum_{k=1}^{L} w_k\right)^2 - \frac{1}{30}\sum_{k=1}^{L} w_k^2.$$

Since $c_3^2 = \frac{1}{36}\left(\sum_{k=1}^{L} w_k\right)^2$, the ratio S is calculated as:

$$S = \frac{c_5}{c_3^2} = \frac{\frac{1}{24}(\sum w_k)^2 - \frac{1}{30}\sum w_k^2}{\frac{1}{36}(\sum w_k)^2} = 1.5 - 1.2\frac{\sum_{k=1}^{L} w_k^2}{\left(\sum_{k=1}^{L} w_k\right)^2}.$$

Define $Q(\mathbf{w}) = \sum_{k=1}^{L} w_k^2 / \left(\sum_{k=1}^{L} w_k\right)^2$. The bounds of S depend entirely on the range of $Q(\mathbf{w})$ for $w_k > 0$.

To find the upper bound of S, we minimize $Q(\mathbf{w})$. By the Cauchy-Schwarz inequality:

$$\left(\sum_{k=1}^{L} w_k\right)^2 \le L\sum_{k=1}^{L} w_k^2.$$

Thus, $Q(\mathbf{w}) \ge 1/L$. Substituting this into S yields the upper bound:

$$S \le 1.5 - \frac{1.2}{L}.$$

Equality holds if and only if $w_1 = w_2 = \cdots = w_L$. Given $w_k = (a_0 \dots a_{k-1})^2$ and $a_0 \neq 0$, this condition is mathematically equivalent to $|a_1| = |a_2| = \cdots = |a_{L-1}| = 1$.

To find the lower bound of S, we maximize $Q(\mathbf{w})$. Expanding the denominator of $Q(\mathbf{w})$ gives:

$$\left(\sum_{k=1}^{L} w_k\right)^2 = \sum_{k=1}^{L} w_k^2 + 2\sum_{1\le j<k\le L} w_j w_k.$$

For the trivial case $L = 1$, the cross-term is zero, yielding exactly $S = 0.3$. However, for any nested structure where $L \ge 2$, since $w_k > 0$, the cross-term is strictly positive, yielding $Q(\mathbf{w}) < 1$ and thus strictly $S > 0.3$. The infimum $S = 0.3$ is approached when a single term dominates the sequence (i.e., $w_m \gg \sum_{j\neq m} w_j$), degenerating the nested structure into an equivalent single sine function. □

B Approximation Details

In this section, we give the exact parameters for our approximations. The nested sine function of depth L is defined as:

$$F_L(x) = a_L \sin\Big(\dots a_1 \sin\big(a_0 x\big)\dots\Big). \tag{32}$$

To derive the optimal parameters for the nested sine approximations, we employ two distinct numerical pipelines depending on the target complexity:

1. Minimax Search (Shallow/Narrow Domains): We directly minimize the maximum absolute error via a two-phase approach. A Differential Evolution algorithm first performs a global search over the bounded parameter space. Its output then serves as the initial guess for a Nelder-Mead simplex algorithm, which locally refines the parameters to a strict tolerance.

2. Progressive TRF Optimization (Deep/Wide Domains): For extreme landscapes, we use non-linear least-squares with a Soft-L_1 loss and a 2-point numerical Jacobian. To ensure convergence, we implement a stage-wise continuation strategy: the Trust-Region-Reflective optimizer is initialised on a narrow central sub-domain. The evaluation interval is then progressively expanded to the full target domain, with the optimized parameters from each stage acting as the initialization for the next.

Tanh Approximation ($L = 4$):
For $x \in [-3.0, 3.0]$ (Max Error: 2.46×10^{-4}):

$$a_0 = 0.475239414002621, \quad a_1 = 1.293688536688026,$$
$$a_2 = 1.223317631122977, \quad a_3 = 1.184219321902110,$$
$$a_4 = 1.121262687622076.$$

For $x \in [-4.0, 4.0]$ (Max Error: 9.55×10^{-4}):

$$a_0 = 0.424050165507937, \quad a_1 = 1.396523122872498,$$
$$a_2 = 1.259795480499946, \quad a_3 = 1.224024134157992,$$
$$a_4 = 1.090421757701098.$$

For $x \in [-5.0, 5.0]$ (Max Error: 1.94×10^{-3}):

$$a_0 = 0.377623276630441, \quad a_1 = 1.521213185100660,$$
$$a_2 = 1.288266813693540, \quad a_3 = 1.250875724362980,$$
$$a_4 = 1.071703022007654.$$

Tanh Approximation ($L = 8$):
For a wider interval $x \in [-30.0, 30.0]$ (Max Error: 1.65×10^{-5}):

$$a_0 = \ 0.087013654658141, \quad a_1 = \ 2.055931586303917,$$
$$a_2 = -1.666801286322939, \quad a_3 = \ 1.058337306942213,$$
$$a_4 = \ 1.392325856907275, \quad a_5 = \ 1.117350327947111,$$
$$a_6 = \ 1.811945957695082, \quad a_7 = \ 0.827748358666983,$$
$$a_8 = -1.357941927687442.$$

GELU Approximation ($L = 3$):
We first approximate the error function component $\mathrm{erf}(x/\sqrt{2}) \approx F_3(x)$, and then compute the GELU function as:

$$\mathrm{GELU}(x) \approx 0.5x\big(1 + F_3(x)\big).$$

For $x \in [-3.0, 3.0]$ (Max Error: 1.20×10^{-4}):

$$a_0 = 0.286561794711382, \quad a_1 = 1.880060125964779,$$
$$a_2 = 1.476727902605978, \quad a_3 = 1.003268674263816.$$

For $x \in [-4.0, 4.0]$ (Max Error: 3.03×10^{-4}):

$$a_0 = 0.313394936087515, \quad a_1 = 1.670106325937232,$$
$$a_2 = 1.524124637183568, \quad a_3 = 1.000884170980229.$$

For $x \in [-5.0, 5.0]$ (Max Error: 6.64×10^{-3}):

$$a_0 = 0.402881881678235, \quad a_1 = 1.651675285056036,$$
$$a_2 = 1.054938655127009, \quad a_3 = 1.148693381934532.$$

GELU Approximation ($L = 8$):
For a wider interval $x \in [-30.0, 30.0]$ (Max Error: 7.90×10^{-4}):

$$a_0 = 0.057642124731843, \quad a_1 = 2.572081612844874,$$
$$a_2 = 2.061709978616492, \quad a_3 = 0.002777346425320,$$
$$a_4 = 3.105511196504953, \quad a_5 = 6.680286494670439,$$
$$a_6 = 29.066529223393925, \quad a_7 = 1.561651045625336,$$
$$a_8 = 1.000068106680754.$$

C Security proofs

C.1 Security of the Attention Scoring Evaluation Protocols

Proof of indistinguishability with corrupted Client. We show that a corrupted Client learns nothing about the Server's private masks or the original values via a hybrid argument.

- **Hyb0:** This corresponds to the real-world distribution where the Client receives the blinded ciphertexts $\mathrm{ct}_C(\hat{Q}), \mathrm{ct}_C(\hat{K})$ and the Server-encrypted masks $\mathrm{ct}_S(M_Q), \mathrm{ct}_S(M_K)$.
- **Hyb1:** The simulator replaces $\mathrm{ct}_S(M_Q)$ and $\mathrm{ct}_S(M_K)$ with encryptions of zero matrices under the Server's key. This is computationally indistinguishable from Hyb0 due to the semantic security of the Server's HE scheme.
- **Hyb2:** After the Client decrypts $\mathrm{ct}_C(\hat{Q})$ and $\mathrm{ct}_C(\hat{K})$ to obtain $\hat{Q}$ and $\hat{K}$, the simulator replaces these blinded plaintexts with uniformly random matrices. Since the masks M_Q and M_K are sampled uniformly at random by the Server and act as one-time pads, the resulting $\hat{Q}$ and $\hat{K}$ are statistically independent of the true Q and K. Thus, Hyb2 is perfectly indistinguishable from Hyb1.

Proof of indistinguishability with corrupted Server. We show that a corrupted Server learns nothing about the Client's intermediate attention scores or private share.

- **Hyb0:** This corresponds to the real-world distribution where the Server receives the ciphertext $\mathsf{C}_{\text{share}}$ from the Client.
- **Hyb1:** The simulator replaces $\mathsf{C}_{\text{share}}$ with an encryption of a uniformly random matrix under the Server's key. During the local computation, the Client subtracts a private, uniformly random secret share S_{client} from the partial results. This S_{client} acts as a one-time pads in the plaintext space, rendering the underlying plaintext of $\mathsf{C}_{\text{share}}$ uniformly random to the Server. Combined with the semantic security of the HE scheme, the Server cannot distinguish the simulated $\mathsf{C}_{\text{share}}$ from the real one.

C.2 Security of the Non-linear Evaluation Protocols

Our non-linear evaluation protocol securely computes the nested sine approximation $F_L(x)$ by sequentially invoking the single-round sine function evaluation protocol over L layers. Since the intermediate results at each layer remain additively secret-shared between the Client and the Server, the security of the overall protocol follows from the sequential composition of the underlying HE-based protocol.

Proof of indistinguishability with corrupted Client. We show that a corrupted Client learns nothing about the Server's secret shares or the true intermediate activation values via a hybrid argument.

- **Hyb0:** This corresponds to the real-world distribution where the Client receives the evaluated ciphertexts $\mathrm{ct}_{y_0}^{(l)}$ from the Server for each layer $l \in \{1, \ldots, L\}$.
- **Hyb1:** For each layer l, the simulator replaces the ciphertext $\mathrm{ct}_{y_0}^{(l)}$ with an encryption of a uniformly random value in $\mathbb{Z}_t$. In the real protocol, the Server blinds the homomorphic evaluation result by subtracting a uniformly sampled secret share $y_1^{(l)} \in \mathbb{Z}_t$. Therefore, the underlying plaintext $y_0^{(l)}$ is perfectly uniform and independent of the actual activation values. Thus, Hyb1 is perfectly indistinguishable from Hyb0, and the Client's view can be simulated without the true data.

Proof of indistinguishability with corrupted Server. We show that a corrupted Server learns nothing about the Client's secret shares or the intermediate activation values.

- **Hyb0:** This corresponds to the real-world distribution where the Server receives the Client's ciphertexts $\mathrm{ct}_{s_0}^{(l)}$ and $\mathrm{ct}_{c_0}^{(l)}$ for each layer $l \in \{1, \ldots, L\}$.
- **Hyb1:** For each layer l, the simulator replaces the Client's ciphertexts with encryptions of zeros. It directly follows from the semantic security (IND-CPA) of the underlying BFV homomorphic encryption scheme that Hyb1 is computationally indistinguishable from Hyb0. Consequently, the Server's view reveals no information about the Client's private data.

C.3 Security of the P2P-based Ciphertext Packaging Protocol

Security properties of our building blocks. Security for our P2P-based ciphertext packaging protocol requires the underlying HE scheme and the RLWE assumption to satisfy the following properties:

- **Semantic security.** For any two messages m, m', we require $\{\mathsf{pk}, \mathsf{HE.Enc}(\mathsf{pk}, m)\} \approx_c \{\mathsf{pk}, \mathsf{HE.Enc}(\mathsf{pk}, m')\}$, where the two distributions are over the random choice of pk and the random coins of the encryption algorithm.
- **Multi-Key Fixed-Base RLWE Pseudo-randomness.** We require that for any uniformly sampled public polynomial $\bar{a}(x) \leftarrow R_q$, and $|\mathcal{U}|$ independently sampled secret keys $s_1(x), \ldots, s_{|\mathcal{U}|}(x)$ and error terms $e_1(x), \ldots, e_{|\mathcal{U}|}(x)$, the concatenated distribution of multiple RLWE samples is computationally indistinguishable from a uniform distribution. That is, $(\bar{a}(x), \bar{a}(x)s_1(x) + e_1(x), \ldots, \bar{a}(x)s_{|\mathcal{U}|}(x) + e_{|\mathcal{U}|}(x)) \approx_c (\bar{a}(x), u_1(x), \ldots, u_{|\mathcal{U}|}(x))$, where $u_k(x) \leftarrow R_q$. This property can be directly reduced to the standard Decisional RLWE assumption via a standard hybrid argument. In our system this property is invoked for the set of public polynomials $\{a_i(x)\}_{i=0}^{d_{in}-1}$ and $a'(x)$.

- **Function privacy.** There exists a simulator Sim_{FP} such that for every efficient adversary $\mathcal{A}$, every linear function $\mathcal{F}$, and input ciphertext ct, the distribution of the homomorphically evaluated ciphertext is computationally indistinguishable from the ciphertext generated by the simulator using only the final output. That is, $\mathsf{HE.Eval}(\mathsf{pk}, \mathsf{ct}, \mathcal{F}) \approx_c \mathrm{Sim}_{FP}(1^\lambda, \mathcal{F}(m))$.

Proof of indistinguishability with corrupted Seeder. We show that the real world distribution is computationally indistinguishable from the simulated distribution via a hybrid argument. In the final simulated distribution, the simulator does not use the real inputs of the users or the server, and so a corrupted Seeder learns nothing beyond the network traffic patterns in the real world.

- **Hyb0:** This corresponds to the real world distribution where the Seeder receives the partial coefficient sets $\{c^{(k)}_{0,j,i} \mid j \in I_k, i \in [0, d_{in} - 1]\}$ from all users in the upstream phase, and receives the blinded ciphertext pair $(\mathrm{ct}'_{0,\mathrm{out},j},\ \mathrm{ct}'_{1,\mathrm{out},j})$ from the server in the downstream phase.
- **Hyb1:** In this hybrid, for the downstream phase, we use the simulator Sim_{FP} to generate the homomorphically evaluated ciphertext pair. It follows from the function privacy of the HE scheme that Hyb1 is computationally indistinguishable from Hyb0.
- **Hyb2:** We change the upstream inputs that the Seeder receives from the users. Instead of providing the actual encrypted coefficients $c^{(k)}_{0,j,i}$, the simulator provides uniformly random coefficients $u_{j,i} \in \mathbb{Z}_q$ for all $j \in I_k$ and all dimensions i. It follows from the Multi-Key Fixed-Base RLWE pseudo-randomness property on $\{a_i(x)\}$ that Hyb2 is computationally indistinguishable from Hyb1.

Proof of indistinguishability with corrupted Server. We show that the real world distribution is computationally indistinguishable from the simulated distribution via a hybrid argument. In the final simulated distribution, the simulator does not use the users' inputs, ensuring that a corrupted server learns nothing about the private data.

- **Hyb0:** This corresponds to the real world distribution where the server receives the aggregated ciphertexts ct_i^{agg} from the Seeder, together with the public keys $\{pk_u\}$, and evaluates the output using its weight matrix W.
- **Hyb1:** In this hybrid, we replace the first component $C^{agg}_{0,i}$ of the aggregated ciphertexts sent to the server. Instead of sending the actual concatenated polynomials constructed from the users' partial ciphertexts, the simulator sends completely random polynomials $U_{agg,i}(x) \in R_q$. The second components $-a_i(x)$ are deterministic public values and carry no user information. It follows from the Multi-Key Fixed-Base RLWE pseudo-randomness on $\{a_i(x)\}$ that the concatenated polynomials are computationally indistinguishable from uniformly random polynomials over R_q. Thus, Hyb1 is computationally indistinguishable from Hyb0.

Proof of indistinguishability with corrupted Leecher. We show that the real world distribution is computationally indistinguishable from the simulated distribution via a hybrid argument. In the final simulated distribution, the simulator does not use the weights of the server's model, and so a corrupted Leecher u_p learns nothing beyond its own output share and the model architecture in the real world.

- **Hyb0:** This corresponds to the real world distribution where the server evaluates the output using its weight matrix W, blinds both ciphertext components using the mask ciphertext constructed from $R_j(x)$, $v_j(x)$, and the user public keys, and broadcasts the blinded pair $(\mathrm{ct}'_{0,\mathrm{out},j},\ \mathrm{ct}'_{1,\mathrm{out},j})$. The Leecher u_p decrypts it using $s_p(x)$.
- **Hyb1:** This hybrid involves only a syntactic change. In the output phase, for the index interval I_p belonging to u_p, the simulator uses the knowledge of the user's input X_p and the server's random tape to directly construct the ciphertext such that its decryption yields $(X_pW)_{I_p} - [R_j]_{I_p} + \epsilon$. Since this is a syntactic change, Hyb1 is distributed identically to Hyb0.
- **Hyb2:** In this hybrid, we change the data in the broadcast ciphertext belonging to other users (i.e., indices $l \notin I_p$). The simulator no longer provides the actual homomorphically evaluated ciphertext components but replaces them with uniformly random values in $\mathbb{Z}_q$. When the corrupted Leecher u_p attempts to decrypt these positions, the result is masked by the term $[\sum_{i=0}^{d_{in}-1} a_i(x)w_{i,j}(s_k(x) - s_p(x))]_l$. It follows from the Decisional RLWE assumption on $\{a_i(x)\}$ that this mask value is computationally indistinguishable from a uniformly random number. Therefore, Hyb2 is computationally indistinguishable from Hyb1.
- **Hyb3:** In this hybrid, we replace the input $(X_pW)_{I_p} - [R_j]_{I_p}$ used to generate the Leecher's decryption share with a randomly sampled value r_p from the plaintext space, and simultaneously replace $\mathrm{ct}'_{1,\mathrm{out},j}$ with a uniformly random polynomial in R_q. The first replacement is identically distributed because the mask $R_j(x)$ sampled by the server is chosen uniformly at random. The second replacement is computationally indistinguishable because $\mathrm{ct}'_{1,\mathrm{out},j} = \sum_{i=0}^{d_{in}-1}(-a_i(x)w_{i,j}) + a'(x) \cdot v_j(x) - e_{s1,j}(x)$, and the term $a'(x) \cdot v_j(x) - e_{s1,j}(x)$ is computationally indistinguishable from uniform under the RLWE assumption on $a'(x)$, which randomizes the entire polynomial. Notice that in this hybrid, the server is no longer using its weight matrix W to generate any component visible to the Leecher. Finally, we note that Hyb3 is identically distributed to the simulator's output, completing the proof.

Security Against Colluding Adversaries. Our protocol inherently resists collusion. Since entities hold asymmetric secrets, any collusion reducible to the following two primary scenarios remains perfectly simulatable:

- **Collusion Against an Honest User (Server + Seeder + Corrupted Leechers):** Even if all other parties collude to pool the Server's weights, Seeder's traffic views, and corrupted Leechers' secret keys, they still lack the honest user's independently generated secret key $s_k(x)$. Under the Multi-Key RLWE assumption (as formalized in our hybrids), the internal HE masking ensures the honest user's data remains perfectly indistinguishable from uniform randomness.

- **Collusion Against the Server (Seeder + All Leechers):** If the Seeder explicitly shares all users' raw inputs X with the Leechers, they can compute the exact upstream plaintexts. However, as shown in Leecher's Hyb3, the final decrypted outputs $(X_p W)_{I_p} - [R_j]_{I_p}$ are strictly blinded by the independent, uniformly random polynomials $R_j(x)$ generated by the Server. This acts as an information-theoretic one-time pad, ensuring their joint view leaks absolutely zero information about the Server's model weights W.