Capricorn: Highly Efficient and Secure Mixture-of-Experts Inference Framework

Lushan Song[1*] **Xiaojian Liang**[1*] **Shishuai Du**[1*] **Jun Jie Sim**[1] **Yingting Liu**[1]
Xin Zhang[1] **Jiang-Ming Yang**[1] **Pu Duan**[1†]
[1]Ant International, Ant Group

Abstract

The rapid advancement of Large Language Models (LLMs) based on Mixture-of-Experts (MoE) architecture has enhanced a growing demand for secure inference frameworks that protect both client inputs and server model weights. However, existing secure MoE inference frameworks suffer from two limitations: (1) a costly two-stage secure routing pipeline that uses expensive secure Top-K and secure equality test protocols, resulting in significant computational and communication overhead, and (2) secure evaluation of nonlinear activations like SiLU requires several multiplication and comparison operations, resulting in significant communication overhead.

In this paper, we present Capricorn, a highly efficient and secure MoE inference framework that overcomes the two limitations above. Firstly, Capricorn utilizes a novel one-stage routing pipeline that reduces both the total number of Top-K protocol invocations and their input data size. At the same time, we propose a lightweight method to generate one-hot token selection matrices locally, thus avoiding expensive secure equality tests. Secondly, we design a secure and precise SiLU protocol, which uses secure lookup tables (LUTs) with dynamic bit-width and precision to reduce communication overhead. Extensive experiments demonstrate that Capricorn achieves up to 40.23× and 1.99× speedup in secure routing and secure SiLU protocols, respectively, compared to the state-of-the-art (SOTA) framework CryptoMoE (NeurIPS'25), while achieving comparable task-level accuracy. Our code is available on GitHub[3].

1 Introduction

Large Language Models (LLMs) based on Mixture-of-Experts (MoE), such as DeepSeek MoE series [4, 17, 18], Qwen MoE series [31, 32], and OLMoE [23], have emerged as a powerful architecture. These models support massive model parameters without incurring a proportional increase in per-token computational cost. However, when deployed in real-world scenarios, these models raise critical privacy concerns [39, 28]. Client inputs typically contain sensitive personal information, while the server model parameters (e.g., expert weights) often constitute valuable intellectual property. For instance, consider a medical diagnosis service where a user submits confidential health records to a cloud-based MoE model for personalized analysis reports. Here, the user would want their personal information to be kept private from the medical service provider, while the service provider would want the model weights that are derived from massive, feature-rich datasets to be kept private, too.

Related Work. A series of works have been proposed [10, 19, 35, 15, 9, 37, 2, 14, 20, 34] to address these privacy concerns. These works primarily use cryptographic techniques such as secure

*Equal contribution.
†Correspondence to p.duan@ant-intl.com.
[3]https://github.com/sls33/Capricorn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

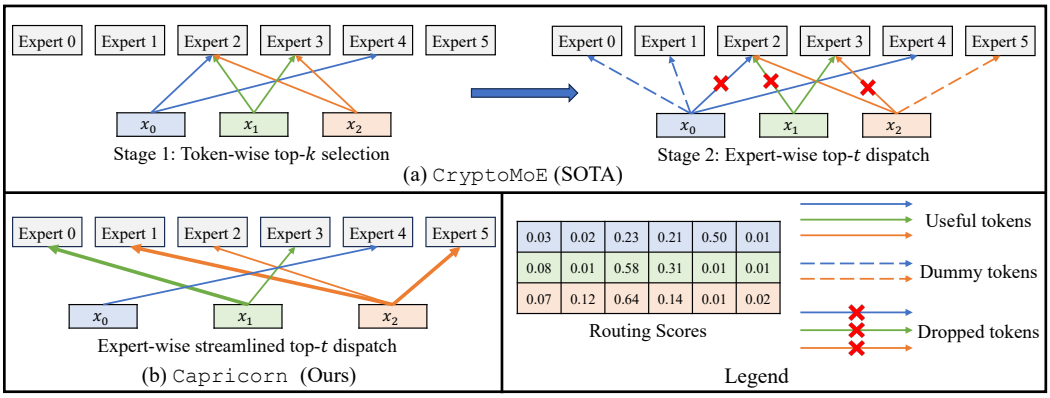

Figure 1: Comparison between CryptoMoE's two-stage routing and Capricorn's modified one-stage routing for $k = 2$ and $t = 1$. Both pipelines start from the same token–expert score matrix and enforce the same per-expert capacity. CryptoMoE first removes token-expert pairs outside each token's top-k set and then performs expert-wise top-t selection over the remaining pairs. Capricorn performs expert-wise top-t selection over all tokens directly. The bold edges in (b) indicate assignments outside the tokens' original top-k sets.

multi-party computation [29, 30, 16, 7, 8] and/or homomorphic encryption [36]. These frameworks provide efficient building blocks for linear layers and non-linear activations, but generally do not address the unique challenge introduced by MoE architectures, i.e., dispatching tokens to n experts while protecting the routing information.

Recent secure MoE frameworks [39, 28] address the MoE-specific challenge through two distinct paradigms. CryptoMoE [39] adopts the "Balanced Routing" paradigm. Every expert participates in the computation and processes a fixed number of tokens (expert capacity), thereby hiding the routing information. This fixed-capacity execution makes the computation and communication independent of the token-expert assignments. SecMoE [28] instead introduces a "Select-Then-Compute" paradigm that securely activates only a subset of experts for each token. Although this paradigm preserves model sparsity, its computational workload scales with the number of activated token-expert assignments. Besides, its HE-based expert evaluation requires costly ciphertext–ciphertext multiplications, whose substantial local computation can dominate even at small token counts.

Limitations of CryptoMoE. In this work, we follow the "Balanced Routing" paradigm. Nevertheless, although providing an effective starting point for secure balanced routing, CryptoMoE [39] has two major efficiency bottlenecks. (1) *Costly two-stage routing pipeline.* As shown in Figure 1(a), CryptoMoE first performs token-wise top-k selection to k experts with the highest routing scores for each token, producing token-expert candidates organized by token. It then regroups these candidates by expert and performs expert-wise top-t dispatch, padding with dummy tokens (zero scores) when fewer than t tokens are available. In secure computation, token-wise top-k and expert-wise top-t require costly Top-K operations, while candidate regrouping and constructing one-hot token-selection matrices for retrieving the selected token require additional secure equality tests. (2) *Expensive secure nonlinear activation evaluations.* CryptoMoE evaluates nonlinear activation (SiLU) using a high-degree polynomial approximation following prior secure transformer framework [19]. Evaluating such a polynomial in secure fixed-point arithmetic requires multiple secure multiplications, truncations and comparisons, resulting in significant communication overhead.

Our Contributions. We propose Capricorn, a highly efficient and secure MoE inference framework to overcome the two limitations above. (1) First, as shown in Figure 1(b), we adopt a modified one-stage routing rule based on expert choice [38]. Rather than first enforcing the token-wise top-k constraint, Capricorn allows each expert to directly select its top-t tokens from all input tokens with public indices. Our key insight is that all experts participate in the dispatch stage anyway, thus token-wise top-k selection is not necessary. The modified rule eliminates the costly token-wise Top-K and secure equality test-based candidate regrouping, while reducing the candidate dimension of the expert-wise dispatch. We further improve efficiency by locally generating one-hot token-selection

matrices based on the public positional indices of the input tokens, avoiding the expensive secure equality tests required in CryptoMoE. (2) Second, we propose a communication-efficient secure SiLU protocol based on secure lookup tables (LUTs) [6]. The SiLU function is divided into several sub-intervals, and each interval is approximated with a linear function. This piecewise linear formulation reduces the multiplications and comparisons required by polynomial-based methods [19, 9]. To further reduce communication overhead, we employ dynamic bit-width and precision for LUTs. As a result, our secure SiLU protocol achieves high accuracy with significantly reduced communication overhead.

Our contributions can be summarized as follows:

- **An efficient one-stage secure routing protocol**. It eliminates token-wise secure Top-K, secure equality test-based candidate regrouping, and secure equality test-based one-hot token selection matrix generation, while reducing the candidate dimension of the expert-wise dispatch.
- **A communication-efficient LUT-based SiLU protocol**. It combines piecewise linear approximations with dynamic bit-width and precision optimization to reduce the number of secure multiplications and comparisons while maintaining high accuracy.

Extensive evaluations over several MoE benchmarks demonstrate that Capricorn significantly outperforms the SOTA framework CryptoMoE (NeurIPS'25), in both communication and runtime. Specifically, our secure routing achieves up to a 40.23× speedup in runtime. Furthermore, our secure SiLU protocol provides a 1.99× speedup in runtime and a 2.46× reduction in communication. As a result, for an MoE layer inference, Capricorn outperforms CryptoMoE by up to 6.43×.

2 Preliminaries

2.1 Notations

In this paper, P_S and P_C denote the server and the client, respectively. For any positive integer n, we use $[n] = \{0, 1, \ldots, n-1\}$ to denote its index set. Furthermore, let N, m, and t denote the number of experts, the number of tokens, and the expert capacity, respectively. Table 1 summarizes the frequently used notations in this paper.

Table 1: Descriptions of notations used in this paper.

Notation	Description
P_S, P_C	The server and the client.
N	The number of experts.
m	The number of tokens.
t	The expert capacity.
T	The lookup table with piecewise linear coefficients a_i, d_i.
$[n]$	The set $\{0, 1, \ldots, n-1\}$.
Π_{Mul}	$\langle z\rangle^\ell = \Pi_{\mathsf{Mul}}(\langle x\rangle^\ell, \langle y\rangle^\ell)$, s.t., $z = x \cdot y$.
Π_{Matmul}	$\langle \mathbf{Z}\rangle^\ell = \Pi_{\mathsf{Matmul}}(\langle \mathbf{X}\rangle, \langle \mathbf{Y}\rangle)$, s.t., $\mathbf{Z} = \mathbf{X} \times \mathbf{Y}$.
Π_{MUX}	$\langle z\rangle^\ell = \Pi_{\mathsf{MUX}}(\langle x\rangle^\ell, \langle b\rangle^1)$, s.t., $z = x$ if $b = 1$, and $z = 0$ if $b = 0$.
Π_{CMP}	$\langle b\rangle^1 = \Pi_{\mathsf{CMP}}(\langle x\rangle^\ell, \langle y\rangle^\ell)$, s.t., $b = 1$ if $x \geq y$, and $b = 0$ otherwise.
Π_{Trunc}	$\langle z\rangle^\ell = \Pi_{\mathsf{Trunc}}(\langle x\rangle^\ell, f)$, s.t., $z = x \gg f$.
Π_{TR}	$\langle z\rangle^{\ell-f} = \Pi_{\mathsf{TR}}(\langle x\rangle^\ell, f)$, s.t., $z = x \gg f$.
Π_{Extend}	$\langle x\rangle^\ell = \Pi_{\mathsf{Extend}}(\langle x\rangle^{\ell'}, \ell)$.
Π_{Topk}	$\langle \mathbf{z}\rangle = \Pi_{\mathsf{Topk}}(\langle \mathbf{x}\rangle, k)$, s.t., $\mathbf{z} = \mathrm{TopK}(\mathbf{x}, k)$.
Π_{LUT}	$\langle z\rangle = \Pi_{\mathsf{LUT}}(T, \langle i\rangle)$, s.t., $z = T[i]$.

2.2 Threat Model

Capricorn considers the two-party private inference between a server P_S, which holds the private model parameters, including the router and expert weights, and a client P_C, which holds the private

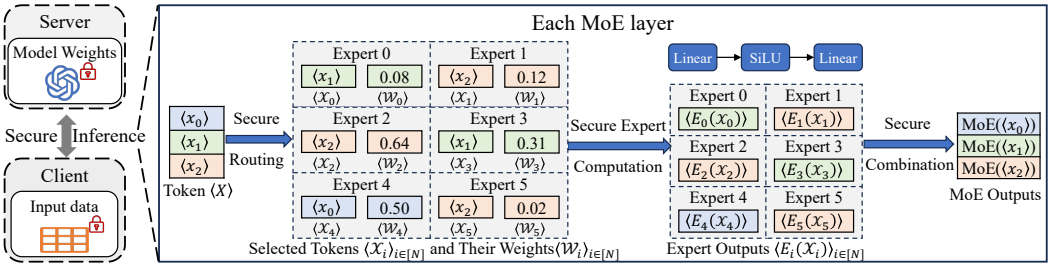

Figure 2: Overview of our proposed Capricorn with a toy example, where the number of tokens $m = 3$, the number of experts $N = 6$, and the expert capacity $t = 1$. $\langle \cdot \rangle$ denotes a secret-shared value. Secure routing, secure expert computation, and secure combination are jointly performed on secret shares.

input. Consistent with previous works [39, 28, 12, 9, 29, 30, 36], we consider a semi-honest threat model, where an adversary could corrupt either the server or the client. Such an adversary follows the protocol specification, but tries to learn more information than allowed.

In Capricorn, all data-dependent routing information, including the private input of each party, routing scores, selected token indices, per-expert one-hot token selection matrices, expert-token assignments, LUT indices are kept secret. The main configurable parameters, including model architecture, tensor shapes, protocol parameters, e.g. (N, m, t, k, ℓ), the LUT T containing the piecewise linear coefficients (a_i, d_i), and the sequence length unless padding is used, are public.

2.3 Mixture of Experts (MoE)

An MoE layer [4, 31, 23] consists of a collection of N independent expert subnetworks $\{E_0, \ldots, E_{N-1}\}$ and a gating network (or a router) G. Given an input x, G computes routing scores through a linear transformation followed by a softmax over all experts: $G(x) = \text{Softmax}(\text{Linear}(x))$. The output of the MoE layer is then obtained as a weighted aggregation of the outputs from the selected experts:

$$\text{MoE}(x) = \sum_{i \in K} \mathbf{W}_i \cdot E_i(x) \tag{1}$$

where K denotes the set of activated experts indices and $\mathbf{W} = \{G(x)_i\}_{i \in K}$ denotes the associated routing scores. Specifically, the architecture is referred to as a dense MoE if $K = \{0, \ldots, N-1\}$, i.e., all experts are activated. Each expert E_i is typically implemented as a feed-forward network (FFN) equipped with nonlinear activation functions such as SiLU. Furthermore, to ensure computational efficiency and load balancing, each expert E_i is typically assigned a fixed capacity t, limiting the maximum number of tokens it can process during a forward pass.

2.4 Cryptographic Primitives

We employ 2-out-of-2 additive secret sharing [5] as the underlying cryptographic primitive. A secret value $x \in \mathbb{Z}_{2^\ell}$ is divided into two arithmetic shares $\langle x \rangle_S^\ell$ (held by P_S) and $\langle x \rangle_C^\ell$ (held by P_C), such that $x = \langle x \rangle_S^\ell + \langle x \rangle_C^\ell \bmod 2^\ell$. In particular, when $\ell = 1$, the addition is equivalent to the XOR operation, i.e., $x = \langle x \rangle_S^1 \oplus \langle x \rangle_C^1 \bmod 2$, and the shares are defined as Boolean shares. For brevity, we omit the superscript ℓ when the bit-width is clear from the context. The secure protocols based on secret sharing used in this paper are summarized in Table 1, and their details are provided in Appendix A.

3 Proposed Method

3.1 Overview

Capricorn consists of three phases for an MoE layer: secure routing, secure expert computation, and secure combination. Figure 2 illustrates an overview of Capricorn with a toy example.

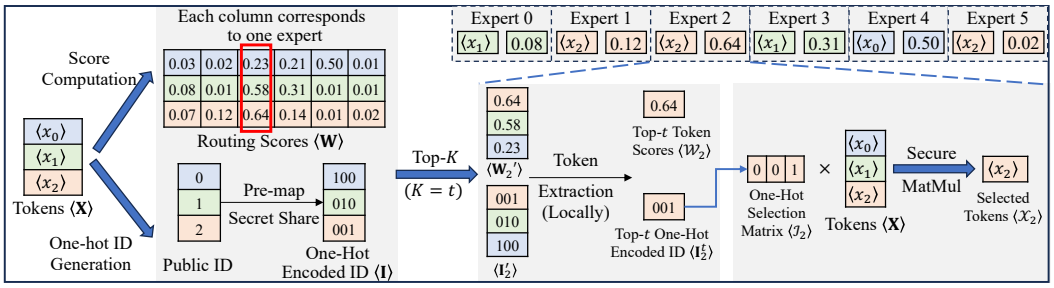

Figure 3: An example of secure routing protocol with $m = 3, N = 6$ and $t = 1$. Score computation, Top-K, and secure MatMul are interactive two-party protocols, while token extraction and one-hot selection matrix conversion are performed locally by each party on its own shares. Candidate token IDs are public, whereas selected token IDs, routing scores, and retrieved tokens remain secret-shared.

- **Secure Routing.** This phase securely dispatches the input tokens $\langle \mathbf{X} \rangle = \{\langle x \rangle_i\}_{i \in [m]} \in \mathbb{Z}^{m \times d}$ to N experts based on the capacity t, according to their routing scores $\langle \mathbf{W} \rangle \in \mathbb{Z}^{m \times N}$. At the end of this phase, each expert E_i receives its dispatched t tokens $\langle \mathcal{X}_i \rangle = \{\langle x \rangle_j | \, x_j \text{ is dispatched to expert } E_i\} \in \mathbb{Z}^{t \times d}$ together with their routing scores $\langle \mathcal{W}_i \rangle \in \mathbb{Z}^t$.
- **Secure Expert Computation.** This phase performs the privacy-preserving evaluation of each expert subnetwork on its dispatched tokens. This phase involves both secure linear operations (e.g., matrix multiplication) and secure non-linear activation (`SiLU` in our implementation) to generate the output $E_i(\langle \mathcal{X}_i \rangle)$ for each expert E_i.
- **Secure Combination.** This phase securely aggregates the expert outputs $E_i(\langle \mathcal{X}_i \rangle)$ for $i \in [N]$, weighted by their corresponding routing scores $\langle \mathcal{W}_i \rangle$, to produce the final token-wise inference outputs $Z \in \mathbb{Z}^{m \times d}$. Our implementation of this phase builds upon the secure combination protocol proposed in `CryptoMoE` [39].

3.2 Secure Routing

The key innovation of our secure routing protocol Π_{Routing} (Protocol 1) lies in two aspects. (1) **Optimized expert-wise streamlined pipeline.** Our protocol allows each expert to directly select its top-t tokens from all input tokens. This eliminates the top-k expert selection and reduces the input data size of the Top-K protocols in the second stage of `CryptoMoE`. (2) **Lightweight method to generate one-hot token selection matrices locally.** Since the token indices are public and fixed, we can pre-map these indices to one-hot encoded representations locally, thus generating the one-hot selection matrices through cheap local operations online. This eliminates the expensive online overhead required for one-hot generation based on the secure equality tests in `CryptoMoE`. These optimizations significantly reduce the computational and communication overhead compared to `CryptoMoE` and enhance the efficiency of secure routing. We provide a toy example with $m = 3, N = 6$ and $t = 1$ in Figure 3 to better illustrate our routing process and provide a detailed description as follows.

P_S and P_C begin by jointly computing the secret shares of the routing scores $\langle \mathbf{W} \rangle \in \mathbb{Z}^{m \times N}$ (Line 1), following the methods in the previous works [39, 21]. Here, each entry $w_{i,j}$ represents the relevance between token $\langle x_i \rangle$ and expert E_j for $i \in [m]$, $j \in [N]$, and $\mathbf{W}_j$ (i.e., the j-th column of $\mathbf{W}$) represents the routing scores associated with expert E_j. At the same time, to avoid the expensive online overhead of secure equality test protocols, the parties locally pre-map the public token indices $\{0, 1, \ldots, m-1\}$ into their corresponding one-hot encoded representations $\mathbf{I} = \{o_0, o_1, \ldots, o_{m-1}\}$, where $o_i \in \{0,1\}^m$. These pre-mapped representations are then secret shared to generate $\langle \mathbf{I} \rangle^1$ (Line 2). Subsequently, for each expert E_i, the parties invoke the secure Top-K protocol Π_{TopK} over the routing scores $\langle \mathbf{W}_i \rangle$ to identify and extract the t highest scores $\langle \mathcal{W}_i \rangle \in \mathbb{Z}^t$ along with their associated one-hot encoded indices $\langle \mathbf{I}_i^t \rangle \in \mathbb{Z}^t$ (Lines 4-5). These indices are then locally converted into a one-hot token selection matrix $\langle \mathcal{I}_i \rangle \in \mathbb{Z}^{t \times m}$ (Line 6), which is used to select t tokens with the highest routing scores. Finally, the parties obtain the dispatched token shares by invoking the secure matrix multiplication protocol Π_{MatMul} between the selection matrix $\langle \mathcal{I}_i \rangle$ and the tokens $\langle \mathbf{X} \rangle$ (Line 7).

Additionally, we analyze the communication complexity of the secure routing protocol in Appendix C.

Protocol 1: Π_{Routing}

Parameters: Number of tokens m; hidden dimension d; number of experts N; expert capacity t.

Input: P_S inputs the private router parameters, and P_S and P_C input their token shares $\langle X \rangle \in \mathbb{Z}^{m \times d}$.

Output: P_S and P_C learn the dispatched token shares $\langle \mathcal{X}_i \rangle \in \mathbb{Z}^{t \times d}$ and the corresponding score shares $\langle \mathcal{W}_i \rangle \in \mathbb{Z}^t$ for each expert E_i ($i \in [N]$).

1: P_S and P_C invoke Π_{Softmax} and Π_{MatMul} to compute the shares of routing scores $\langle \mathbf{W} \rangle \leftarrow \{G(\langle \mathbf{X} \rangle)_i\}_{i \in [N]} \in \mathbb{Z}^{m \times N}$, where $G(\mathbf{X}) = \texttt{Softmax}(\text{Linear}(\mathbf{X}))$.
2: P_S and P_C pre-map the token indices $\{0, \ldots, m-1\}$ to their corresponding one-hot encoded representations $\mathbf{I} = \{o_0, o_1, \ldots, o_{m-1}\}$, where $o_i \in \{0,1\}^m$, and then generate the shares $\langle \mathbf{I} \rangle^1$.
3: **for** each expert E_i for $i \in [N]$ **do**
4: P_S and P_C invoke $\Pi_{\mathsf{Topk}}(\langle \mathbf{W}_i \rangle, t)$ to learn the shares of sorted scores $\langle \mathbf{W}'_i \rangle \in \mathbb{Z}^m$ and the corresponding one-hot encoded token indices $\langle \mathbf{I}'_i \rangle \in \mathbb{Z}^m$.
5: P_S and P_C locally extract the top-t token scores $\langle \mathcal{W}_i \rangle \in \mathbb{Z}^t$ and the corresponding one-hot encoded token indices $\langle \mathbf{I}'_i \rangle \in \mathbb{Z}^t$.
6: P_S and P_C convert $\langle \mathbf{I}'_i \rangle \in \mathbb{Z}^t$ to its corresponding one-hot token selection matrix $\langle \mathcal{I}_i \rangle \in \mathbb{Z}^{t \times m}$.
7: P_S and P_C invoke $\Pi_{\mathsf{MatMul}}(\langle \mathcal{I}_i \rangle, \langle X \rangle)$ to learn the dispatched tokens $\langle \mathcal{X}_i \rangle \in \mathbb{Z}^{t \times d}$.
8: **end for**

3.3 Secure SiLU

We adopt $\texttt{SiLU}$ as the activation function in the MoE layers. It is defined as $\texttt{SiLU}(x) = x \cdot \texttt{Sigmoid}(x)$, where $\texttt{Sigmoid}(x) = \frac{1}{1+e^{-x}}$. To efficiently evaluate $\texttt{SiLU}(x)$ with privacy preservation, we divide it into three segments based on the boundary points α and β ($\alpha < 0 < \beta$):

$$\texttt{SiLU}(x) \approx \begin{cases} 0, & x < \alpha \\ F(x), & \alpha \le x < \beta \\ x, & x \ge \beta \end{cases} \tag{2}$$

The first and third segments are straightforward to compute. For the second segment, we further divide the interval $[\alpha, \beta)$ into 2^s smaller sub-intervals, each of which is approximated by a linear function $y_i(x) = a_i x + d_i$ as in Equation (3).

$$F(x) \approx \begin{cases} y_0(x), & \alpha \le x < \beta_1 \\ y_1(x), & \beta_1 \le x < \beta_2 \\ \ldots & \\ y_{2^s-1}(x), & \beta_{2^s-1} \le x < \beta \end{cases} \tag{3}$$

Our proposed secure $\texttt{SiLU}$ protocol Π_{SiLU} is described in Protocol 2. We assume that the input x and all linear parameters (a_i, d_i) for $i \in [2^s]$ have been encoded as fixed-point numbers, and that linear parameters have been stored in a public table T. Let the total bit-width and fractional precision of input x be ℓ and f, respectively, while those of each slope a_i and intercept d_i are ℓ_a, f_a and ℓ_d, f, respectively. Inspired by the optimized method for secure $\texttt{GeLU}$ computation in CipherGPT [11], we adopt a secure LUT protocol to improve the efficiency of $\texttt{SiLU}$ evaluation. To further improve efficiency, we use a reduced bit-width for slope a_i and intercept d_i, since these parameters typically lie in the range $(-2, 2)$ (see details in Appendix B). Specifically, we set $\ell_a = f_a + 2$ and $\ell_d = f + 2$. This bit-width optimization significantly reduces the communication size associated with the LUT.

We first describe the case for $x \in [\alpha, \beta)$ (Lines 1-8). Since directly checking which sub-interval x falls into would require an extra sign check step, we shift the input by subtracting α, i.e., $x' = x - \alpha$

(Line 1). After this shift, the second segment becomes $[0, \beta - \alpha)$. Then we only need to consider the lower $L = \log(\beta - \alpha)$ bits of x' to obtain the index that the sub-interval x' falls into. Suppose that $\beta - \alpha$ is a power of two, P_S and P_C can locally extract the lower L bits of x' (Line 2). The exact sub-interval index is found by truncating the lower $(L-s)$ bits via the truncation and reduction protocol $\Pi_{\mathsf{TR}}(\langle x' \rangle^L, L-s)$ (Line 3). With the index, P_S and P_C invoke the LUT protocol to securely retrieve the associated linear parameters a_i and d_i (Line 4). Before computing the linear function $y = a_i x + d_i$, P_S and P_C extend the bit-width of a_i and d_i from ℓ_a and ℓ_d to ℓ, respectively, to match the bit-width of input x (Line 5). Then they securely compute the product $a_i x$ via the secure multiplication protocol Π_{Mul} (Line 6). Since a_i has a lower fractional precision f_a than the input, we truncate the product by f_a bits to align the decimal points (Line 7). Finally, P_S and P_C add the truncated result with d_i to get the result of the linear function $y = a_i x + d_i$ for the second segment (Line 8).

To combine the results from all three segments of the `SiLU` function, we use a three-way conditional selection (Lines 9-12). First, two secure comparison protocols Π_{CMP} are performed to determine which segment the input x belongs to (Lines 9-10). One returns a flag b_0 that indicates if $x' \geq 0$, and another returns a flag b_1 that indicates if $x' \geq \beta - \alpha$. Then, two secure multiplexer protocols Π_{MUX} are performed. The first multiplexer uses $b = b_0 \oplus b_1$ to select between the result of the second segment (y) and the other segment (Line 11). The second multiplexer uses b_1 to select between the result of the first multiplexer and the third segment (x) (Lines 12). The final output z of `SiLU` is obtained by a local addition (Line 13).

Protocol 2: Π_{SiLU}

Parameters: The total bit-width and fractional precision of input x are ℓ and f; the total bit-width and fractional precision of a_i are ℓ_a and f_a, while those of d_i are ℓ_d and f; splitting points α and β, where $\alpha < 0 < \beta$;

Input: A public table T containing the 2^s pairs of piecewise linear coefficients (a_j, d_j); P_S and P_C input their data shares $\langle x \rangle^\ell$ with f-bit precision.

Output: P_S and P_C learn $\langle z \rangle^\ell$, where $z \approx \mathtt{SiLU}(x)$.

1: P_S and P_C locally compute $\langle x' \rangle^\ell \leftarrow \langle x \rangle^\ell - \alpha$.
2: P_S and P_C set $\langle x' \rangle^L \leftarrow \langle x' \rangle^\ell \bmod 2^L$, where $L = \log(\beta - \alpha)$.
3: P_S and P_C invoke $\Pi_{\mathsf{TR}}(\langle x' \rangle^L, L-s)$ to learn the secret-shared segment index $\langle i \rangle^s$.
4: P_S and P_C invoke $\Pi_{\mathsf{LUT}}(T, \langle i \rangle^s)$ to learn the linear function parameters $(\langle a_i \rangle^{\ell_a}, \langle d_i \rangle^{\ell_d})$.
5: P_S and P_C invoke $\Pi_{\mathsf{Extend}}(\langle a_i \rangle^{\ell_a}, \ell)$ and $\Pi_{\mathsf{Extend}}(\langle d_i \rangle^{\ell_d}, \ell)$ to learn $\langle a_i \rangle^\ell$ and $\langle d_i \rangle^\ell$, respectively.
6: P_S and P_C invoke $\Pi_{\mathsf{Mul}}(\langle a_i \rangle^\ell, \langle x \rangle^\ell)$ to learn $\langle a_i x \rangle^\ell$.
7: P_S and P_C invoke $\Pi_{\mathsf{Trunc}}(\langle a_i x \rangle^\ell, f_a)$ to learn $\langle r \rangle^\ell$, such that $r = a_i x \gg f_a$.
8: P_S and P_C locally compute $\langle y \rangle^\ell \leftarrow \langle r \rangle^\ell + \langle d_i \rangle^\ell$.
9: P_S and P_C invoke $\Pi_{\mathsf{CMP}}(\langle x' \rangle^\ell, 0)$ to learn $\langle b_0 \rangle^1$, where $b_0 = 1$ if $x' \geq 0$ and $b_0 = 0$ otherwise.
10: P_S and P_C invoke $\Pi_{\mathsf{CMP}}(\langle x' \rangle^\ell, \beta - \alpha)$ to learn $\langle b_1 \rangle^1$, where $b_1 = 1$ if $x' \geq \beta - \alpha$ and $b_1 = 0$ otherwise.
11: P_S and P_C invoke $\Pi_{\mathsf{MUX}}(\langle y \rangle^\ell, \langle b \rangle^1)$ to learn $\langle e \rangle^\ell$, where $b = b_0 \oplus b_1$.
12: P_S and P_C invoke $\Pi_{\mathsf{MUX}}(\langle x \rangle^\ell, \langle b_1 \rangle^1)$ to learn $\langle u \rangle^\ell$.
13: P_S and P_C locally compute $\langle z \rangle^\ell \leftarrow \langle e \rangle^\ell + \langle u \rangle^\ell$.

4 Experiments

4.1 Experimental Setup

Implementation. We implemented `Capricorn` in C++, building upon the open-source code of SPU[4] [21]. By default, we set the total bit-width $\ell = 64$ with a fractional precision of $f = 18$. For the specific evaluation of `SiLU`, we adjust the total bit-width to $\ell = 40$. At the same time, the splitting points are defined as $\alpha = -8 \cdot 2^f$ and $\beta = 8 \cdot 2^f$. For the piecewise linear parameters, we set the

[4] `https://github.com/secretflow/spu`

total bit-width and fractional precision of each slope a_i to $\ell_a = 14$ and $f_a = 12$, and those for each intercept d_i to $\ell_d = 20$ and $f = 18$. Following CryptoMoE, the capacity factor is fixed to 2.0.

Environment. All of our experiments are conducted on a server equipped with two AMD EPYC 9554 CPUs (128 cores, 3.1GHz) and 1.1 TB RAM. To simulate different network conditions, we use the traffic control command in Linux to manipulate the bandwidth and latency. Specifically, we consider two network environments: (1) a LAN setting with 3 Gbps bandwidth and 0.2ms Round-Trip Time (RTT), and (2) a WAN setting with 400 Mbps bandwidth and 40ms RTT. Experimental results are reported as means over five runs, with sample standard deviations shown as subscripts.

Models and Datasets. Similar to CryptoMoE, we applied Capricorn on three representative MoE models: DeepSeekMoE (`deepseek-ai/deepseek-moe-16b-base`), QwenMoE (`Qwen/Qwen1.5-MoE-A2.7B`), and OLMoE (allenai/OLMoE-1B-7B-0125-Instruct), and evaluated them on six widely used datasets, i.e., OpenBookQA (OBQA) [22], ARC-easy [3], ARC-challenge [3], HellaSwag [33], PIQA [1], and Winogrande [27].

Baselines. Since the authors of CryptoMoE [39] had not yet open-sourced their ciphertext-based implementation before our paper was accepted, we consider two baselines for efficiency evaluation: $\texttt{CryptoMoE}_{\text{serial}}$ and $\texttt{CryptoMoE}_{\text{parallel}}$. These variants perform expert-wise top-t selection sequentially and in parallel, respectively, while preserving the same routing rule and polynomial SiLU approximation.

4.2 Routing Rule Difference

As illustrated in Figure 1, Capricorn adopts a modified one-stage routing rule, rather than an exact or mathematically equivalent replacement for CryptoMoE 's two-stage routing. We take DeepSeekMoE with a batch size of 64 as an example to directly show the differences across six datasets between the two routing rules. These measurements are performed in plaintext with floats.

Let $\mathcal{A}_{\text{CryptoMoE}}$ and $\mathcal{A}_{\texttt{Capricorn}}$ denote the expert-token assignments produced across all routed layers and tokens. Routing overlap measures the fraction of CryptoMoE assignments that are also retained by Capricorn, i.e., $\frac{|A_{\texttt{CryptoMoE}} \cap A_{\texttt{Capricorn}}|}{|A_{\texttt{CryptoMoE}}|}$. Outside top-$k$ measures the fraction of Capricorn assignments for which the assigned expert does not belong to the corresponding token's top-k candidate set under CryptoMoE. As shown in Table 2, Capricorn retains 98.20% of CryptoMoE assignments, while 53.10% of Capricorn assignments fall outside the original top-k. These percentages use different denominators and show that Capricorn mainly adds new expert paths while retaining the original ones. At the output level, the logit relative ℓ_2 error is 0.053, with a Top-1 agreement of 92.80%. Thus, Capricorn preserves most CryptoMoE assignments but is neither assignment-equivalent nor output-equivalent.

Table 2: Routing rule difference between one-stage routing and two-stage routing.

Dataset	Routing overlap	Outside Top-k	Logit relative ℓ_2	Top-1 agreement
OpenBookQA	96.96%	54.63%	0.066	91.47%
ARC-Easy	97.50%	53.90%	0.065	91.24%
ARC-Challenge	97.74%	53.66%	0.064	91.90%
HellaSwag	98.34%	52.90%	0.051	92.92%
PIQA	97.25%	55.32%	0.059	92.15%
WinoGrande	97.07%	53.08%	0.057	92.76%
Micro aggregate	**98.20%**	**53.10%**	**0.053**	**92.80%**

4.3 Performance Evaluation

4.3.1 Evaluation of Secure Routing

We evaluate the communication and runtime of our secure routing protocol Π_{Routing} against both the serial and parallel implementations of CryptoMoE [39] on the DeepSeekMoE model. Table 3 reports the comparison results under LAN and WAN settings for different numbers of input tokens. Our protocol consistently outperforms both implementations. Compared with $\texttt{CryptoMoE}_{\text{serial}}$, it reduces

communication by $1.09\times \sim 1.18\times$ and achieves runtime speedups of $2.64\times \sim 3.04\times$ in LAN and $30.09\times \sim 40.23\times$ in WAN. Compared with $\texttt{CryptoMoE}_{\text{parallel}}$, our protocol reduces communication by $1.14\times \sim 1.35\times$ while remaining $2.01\times \sim 2.22\times$ faster in LAN and $16.54\times \sim 17.54\times$ faster in WAN. These improvements mainly stem from the two efficiency optimizations in our secure routing protocol analyzed in Section 3.2.

Table 3: Evaluation of secure routing.

Tokens	Framework	Communication (MiB)	Runtime (s)	
			LAN	WAN
16	$\texttt{CryptoMoE}_{\text{serial}}$	$96.86_{\pm 0.797}$	$16.18_{\pm 0.951}$	$673.37_{\pm 89.218}$
	$\texttt{CryptoMoE}_{\text{parallel}}$	$100.97_{\pm 2.146}$	$11.79_{\pm 0.332}$	$370.15_{\pm 8.758}$
	Ours	$88.55_{\pm 0.001}$	$5.32_{\pm 0.035}$	$22.38_{\pm 0.179}$
32	$\texttt{CryptoMoE}_{\text{serial}}$	$141.39_{\pm 1.991}$	$27.61_{\pm 1.793}$	$964.47_{\pm 61.570}$
	$\texttt{CryptoMoE}_{\text{parallel}}$	$151.33_{\pm 2.975}$	$20.61_{\pm 0.631}$	$512.39_{\pm 15.248}$
	Ours	$124.73_{\pm 0.002}$	$10.23_{\pm 0.498}$	$30.32_{\pm 0.237}$
64	$\texttt{CryptoMoE}_{\text{serial}}$	$233.24_{\pm 2.482}$	$50.60_{\pm 3.634}$	$1764.40_{\pm 402.292}$
	$\texttt{CryptoMoE}_{\text{parallel}}$	$267.01_{\pm 13.264}$	$39.64_{\pm 2.606}$	$769.38_{\pm 15.125}$
	Ours	$197.89_{\pm 0.002}$	$19.17_{\pm 0.128}$	$43.86_{\pm 0.439}$

4.3.2 Evaluation of Secure SiLU

We evaluate our secure SiLU protocol Π_{SiLU} against the polynomial approximation method [19] used in CryptoMoE [39]. Table 4 presents the comparison results for input vectors of lengths 2^{18}, 2^{20}, and 2^{22}. The results demonstrate that our protocol reduces communication by $2.41\times \sim 2.46\times$, and achieves runtime speedups of up to $1.99\times$ under WAN and $1.13\times$ under LAN. We also evaluate approximation accuracy using the mean absolute error (MAE), computed as the average absolute difference between the exact and approximated results. Across the three tested input ranges, our protocol achieves a $2.73\times \sim 2.98\times$ reduction in MAE compared with CryptoMoE.

Table 4: Evaluation of secure SiLU.

Length	Framework	Communication (MiB)	Runtime (s)		MAE ($\times 10^{-3}$)		
			LAN	WAN	$[-8,8]$	$[-32,32]$	$[-64,64]$
2^{18}	CryptoMoE	$237.15_{\pm 0.006}$	$6.87_{\pm 0.172}$	$15.96_{\pm 0.127}$	$8.34_{\pm 0.033}$	$2.19_{\pm 0.009}$	$1.09_{\pm 0.009}$
	Ours	$96.42_{\pm 0.000}$	$6.41_{\pm 0.457}$	$8.13_{\pm 0.711}$	$2.81_{\pm 0.016}$	$0.80_{\pm 0.008}$	$0.40_{\pm 0.006}$
2^{20}	CryptoMoE	$880.52_{\pm 0.012}$	$12.91_{\pm 0.335}$	$33.75_{\pm 0.169}$	$8.36_{\pm 0.011}$	$2.19_{\pm 0.003}$	$1.09_{\pm 0.006}$
	Ours	$358.94_{\pm 0.000}$	$11.40_{\pm 0.746}$	$16.95_{\pm 0.719}$	$2.81_{\pm 0.005}$	$0.80_{\pm 0.003}$	$0.40_{\pm 0.002}$
2^{22}	CryptoMoE	$3453.99_{\pm 0.024}$	$38.43_{\pm 1.247}$	$111.69_{\pm 0.427}$	$8.35_{\pm 0.005}$	$2.18_{\pm 0.007}$	$1.10_{\pm 0.001}$
	Ours	$1434.32_{\pm 0.001}$	$34.15_{\pm 2.206}$	$57.91_{\pm 1.640}$	$2.81_{\pm 0.003}$	$0.80_{\pm 0.001}$	$0.40_{\pm 0.001}$

4.4 Evaluation of an MoE Layer

We evaluate the end-to-end runtime of a single MoE layer in DeepSeekMoE under LAN and WAN settings. As illustrated in Figure 4, Capricorn outperforms both $\texttt{CryptoMoE}_{\text{serial}}$ and $\texttt{CryptoMoE}_{\text{parallel}}$ across all tested numbers of input tokens. Under LAN settings, Capricorn achieves speedups of $1.23\times \sim 1.29\times$ over $\texttt{CryptoMoE}_{\text{serial}}$ and approximately $1.15\times$ over $\texttt{CryptoMoE}_{\text{parallel}}$. The improvement is more pronounced under WAN settings, where secure routing accounts for approximately $83\% \sim 86\%$ and $68\% \sim 78\%$ of the total runtime of $\texttt{CryptoMoE}_{\text{serial}}$ and $\texttt{CryptoMoE}_{\text{parallel}}$, respectively. In this setting, Capricorn achieves speedups of $5.47\times \sim 6.43\times$ and $2.91\times \sim 3.93\times$ over the two variants, respectively.

To evaluate the benefits beyond a single MoE layer, we additionally conduct an end-to-end evaluation on Google V-MoE [26] (Appendix D.2).

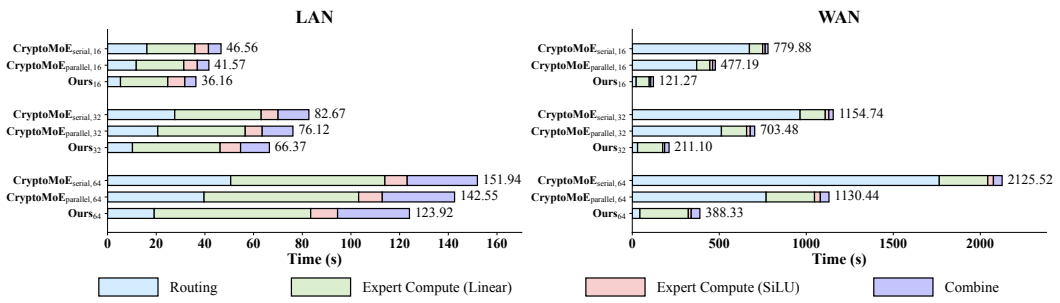

Figure 4: Evaluation of an MoE layer. The subscript of the framework name indicates the number of input tokens.

4.5 Overall Model Accuracy Evaluation

We evaluate a 2×3 factorial on three MoE models to assess the accuracy effects of routing and `SiLU` approximation. The design combines two routing rules (`CryptoMoE`'s two-stage routing and `Capricorn`'s modified one-stage routing) with three activation implementations (exact `SiLU`, `CryptoMoE`'s polynomial approximation, and `Capricorn`'s piecewise-linear approximation). The two-stage/polynomial and one-stage/piecewise-linear configurations correspond to `CryptoMoE` and `Capricorn`, respectively. All approximate-`SiLU` configurations are evaluated under fixed-point emulation.

Table 5 reports accuracy averaged over the six datasets for batch sizes 16 and 64. Detailed per-dataset results are provided in Appendix D.1. Across all models and batch sizes, differences in average accuracy among the six routing-activation combinations are smaller than the reported standard errors. These results demonstrate that `Capricorn` achieves task-level accuracy comparable to the baselines under the evaluated settings.

Table 5: Average accuracy over six datasets. Subscripts denote the arithmetic mean of the six per-dataset standard errors.

Model	Batch	Routing	Exact SiLU	Polynomial	Piecewise-linear
DeepSeekMoE	16	Two-stage	$64.31\%_{\pm 1.20\%}$	$64.49\%_{\pm 1.21\%}$	$64.30\%_{\pm 1.21\%}$
		One-stage	$64.81\%_{\pm 1.20\%}$	$64.46\%_{\pm 1.21\%}$	$64.71\%_{\pm 1.21\%}$
	64	Two-stage	$64.60\%_{\pm 1.21\%}$	$64.62\%_{\pm 1.20\%}$	$64.61\%_{\pm 1.21\%}$
		One-stage	$64.85\%_{\pm 1.20\%}$	$65.02\%_{\pm 1.21\%}$	$64.65\%_{\pm 1.21\%}$
QwenMoE	16	Two-stage	$63.61\%_{\pm 1.21\%}$	$63.77\%_{\pm 1.21\%}$	$63.57\%_{\pm 1.21\%}$
		One-stage	$63.80\%_{\pm 1.21\%}$	$64.08\%_{\pm 1.21\%}$	$64.32\%_{\pm 1.21\%}$
	64	Two-stage	$63.66\%_{\pm 1.21\%}$	$64.10\%_{\pm 1.21\%}$	$64.13\%_{\pm 1.21\%}$
		One-stage	$64.05\%_{\pm 1.21\%}$	$64.24\%_{\pm 1.21\%}$	$64.41\%_{\pm 1.21\%}$
OLMoE	16	Two-stage	$65.58\%_{\pm 1.21\%}$	$65.51\%_{\pm 1.22\%}$	$65.02\%_{\pm 1.22\%}$
		One-stage	$65.86\%_{\pm 1.21\%}$	$66.05\%_{\pm 1.21\%}$	$66.06\%_{\pm 1.21\%}$
	64	Two-stage	$65.74\%_{\pm 1.21\%}$	$65.92\%_{\pm 1.21\%}$	$65.82\%_{\pm 1.21\%}$
		One-stage	$66.04\%_{\pm 1.21\%}$	$66.52\%_{\pm 1.21\%}$	$65.92\%_{\pm 1.21\%}$

5 Conclusion

In this paper, we propose `Capricorn`, a highly efficient secure inference framework for MoE-based LLMs. We propose a novel secure routing protocol with an optimized expert-wise streamlined pipeline and a communication-efficient LUT-based `SiLU` protocol to improve the efficiency of the SOTA secure MoE inference framework `CryptoMoE` [39]. The experimental results demonstrate that `Capricorn` outperforms `CryptoMoE` in both communication and runtime.

References

[1] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In Proceedings of the AAAI conference on artificial intelligence, volume 34, pages 7432–7439, 2020.

[2] Yuntian Chen, Zhanyong Tang, Tianpei Lu, Bingsheng Zhang, Zhiying Shi, and Zheng Wang. Accelerating private large transformers inference through fine-grained collaborative computation. IEEE Transactions on Information Forensics and Security, 2025.

[3] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv preprint arXiv:1803.05457, 2018.

[4] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1280–1297, 2024.

[5] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In Proceedings 2015 Network and Distributed System Security Symposium. Internet Society, 2015.

[6] Ghada Dessouky, Farinaz Koushanfar, Ahmad-Reza Sadeghi, Thomas Schneider, Shaza Zeitouni, and Michael Zohner. Pushing the communication barrier in secure computation using lookup tables. In Proceedings 2017 Network and Distributed System Security Symposium. Internet Society, 2017.

[7] Ye Dong, Chen Xiaojun, Weizhan Jing, Li Kaiyun, and Weiping Wang. Meteor: Improved secure 3-party neural network inference with reducing online communication costs. In Proceedings of the ACM Web Conference 2023, WWW '23, page 2087–2098, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394161. doi: 10.1145/3543507.3583272. URL https://doi.org/10.1145/3543507.3583272.

[8] Ye Dong, Wen-jie Lu, Xiaoyang Hou, Kang Yang, and Jian Liu. M&m: Secure two-party machine learning through modulus conversion and mixed-mode protocols. IEEE Transactions on Dependable and Secure Computing, 2025.

[9] Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, Wen-Guang Chen, et al. Puma: Secure inference of llama-7b in five minutes. Security and Safety, 4:2025014, 2025.

[10] Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. Advances in neural information processing systems, 35: 15718–15731, 2022.

[11] Xiaoyang Hou, Jian Liu, Jingyu Li, Yuhan Li, Jiawen Zhang, Wen-jie Lu, Cheng Hong, and Kui Ren. Ciphergpt: Secure two-party gpt inference. IEEE Transactions on Dependable and Secure Computing, 23(03):6748–6762, 2026.

[12] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. Cheetah: Lean and fast secure {Two-Party} deep neural network inference. In 31st USENIX security symposium (USENIX Security 22), pages 809–826, 2022.

[13] Kristjän Valur Jönsson, Gunnar Kreitz, and Misbah Uddin. Secure multi-party sorting and applications. Cryptology ePrint Archive, 2011.

[14] Andes Y. L. Kei and Sherman S. M. Chow. SHAFT: secure, handy, accurate and fast transformer inference. In 32nd Annual Network and Distributed System Security Symposium, NDSS, 2025.

[15] Zhengyi Li, Kang Yang, Jin Tan, Wen-jie Lu, Haoqi Wu, Xiao Wang, Yu Yu, Derun Zhao, Yancheng Zheng, Minyi Guo, et al. Nimbus: Secure and efficient two-party inference for transformers. Advances in Neural Information Processing Systems, 37:21572–21600, 2024.

[16] Xiaojian Liang, Lushan Song, Shishuai Du, Weicheng Zhu, Tan Li Hui Faith, Jun Jie Sim, Haibing Jin, Zhenghao Wu, Yingting Liu, Xin Zhang, et al. Pisces: Cryptography-based private retrieval-augmented generation with dual-path retrieval. In The Fourteenth International Conference on Learning Representations, 2026.

[17] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. arXiv preprint arXiv:2405.04434, 2024.

[18] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437, 2024.

[19] Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. Bumblebee: Secure two-party inference framework for large transformers. In 32nd Annual Network and Distributed System Security Symposium, NDSS, 2025.

[20] Jinglong Luo, Yehong Zhang, Zhuo Zhang, Jiaqi Zhang, Xin Mu, Hui Wang, Yue Yu, and Zenglin Xu. Secformer: Fast and accurate privacy-preserving inference for transformer models via smpc. In Findings of the Association for Computational Linguistics: ACL 2024, pages 13333–13348, 2024.

[21] Junming Ma, Yancheng Zheng, Jun Feng, Derun Zhao, Haoqi Wu, Wenjing Fang, Jin Tan, Chaofan Yu, Benyu Zhang, and Lei Wang. Secretflow-spu: A performant and user-friendly framework for privacy-preserving machine learning. In 2023 USENIX annual technical conference (USENIX ATC 23), pages 17–33, 2023.

[22] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Proceedings of the 2018 conference on empirical methods in natural language processing, pages 2381–2391, 2018.

[23] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. arXiv preprint arXiv:2409.02060, 2024.

[24] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow2: Practical 2-party secure inference. In Proceedings of the 2020 ACM SIGSAC conference on computer and communications security, pages 325–342, 2020.

[25] Deevashwer Rathee, Mayank Rathee, Rahul Kranti Kiran Goli, Divya Gupta, Rahul Sharma, Nishanth Chandran, and Aseem Rastogi. Sirnn: A math library for secure rnn inference. In 2021 IEEE Symposium on Security and Privacy (SP), pages 1003–1020. IEEE, 2021.

[26] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, Advances in Neural Information Processing Systems, volume 34, pages 8583–8595. Curran Associates, Inc., 2021.

[27] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. Communications of the ACM, 64(9):99–106, 2021.

[28] Bowen Shen, Yuyue Chen, Peng Yang, Bin Zhang, Xi Zhang, and Zoe L Jiang. Secmoe: Communication-efficient secure moe inference via select-then-compute. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 25286–25294, 2026.

[29] Lushan Song, Jiaxuan Wang, Zhexuan Wang, Xinyu Tu, Guopeng Lin, Wenqiang Ruan, Haoqi Wu, and Weili Han. Pmpl: A robust multi-party learning framework with a privileged party. In Proceedings of the 2022 ACM SIGSAC conference on computer and communications security, pages 2689–2703, 2022.

[30] Lushan Song, Zhexuan Wang, Guopeng Lin, and Weili Han. Ruyi: A configurable and efficient secure multi-party learning framework with privileged parties. IEEE Transactions on Information Forensics and Security, 19:10355–10370, 2024.

[31] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. arXiv preprint arXiv:2407.10671, 2024.

[32] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

[33] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Proceedings of the 57th annual meeting of the association for computational linguistics, pages 4791–4800, 2019.

[34] Chenkai Zeng, Debiao He, Qi Feng, Xiaolin Yang, and Qingcai Luo. Securegpt: A framework for multi-party privacy-preserving transformer inference in gpt. IEEE Transactions on Information Forensics and Security, 19:9480–9493, 2024.

[35] Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen-jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. Secure transformer inference made non-interactive. In 32nd Annual Network and Distributed System Security Symposium, NDSS, 2025.

[36] Linru Zhang, Xiangning Wang, Jun Jie Sim, Zhicong Huang, Jiahao Zhong, Huaxiong Wang, Pu Duan, and Kwok-Yan Lam. Moai: Module-optimizing architecture for non-interactive secure transformer inference. In The Fourteenth International Conference on Learning Representations, 2026.

[37] Yancheng Zhang, Jiaqi Xue, Mengxin Zheng, Mimi Xie, Mingzhe Zhang, Lei Jiang, and Qian Lou. Cipherprune: Efficient and scalable private transformer inference. In The Thirteenth International Conference on Learning Representations, 2025.

[38] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. Advances in Neural Information Processing Systems, 35:7103–7114, 2022.

[39] Yifan Zhou, Tianshi Xu, Jue Hong, Ye Wu, and Meng Li. Cryptomoe: Privacy-preserving and scalable mixture of experts inference via balanced expert routing. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

A Secret Sharing-based Protocols

Secure Multiplication. The secure multiplication protocol Π_{Mul} [21] takes arithmetic shares $\langle x\rangle^{\ell}$ and $\langle y\rangle^{\ell}$ as input and returns $\langle z\rangle^{\ell}$, such that $z = x \cdot y$.

Secure Matrix Multiplication. The secure matrix multiplication protocol Π_{MatMul} [21] takes matrix arithmetic shares $\langle \mathbf{X}\rangle^{\ell}$ and $\langle \mathbf{Y}\rangle^{\ell}$ as input and returns $\langle \mathbf{Z}\rangle^{\ell}$, such that $\mathbf{Z} = \mathbf{X} \times \mathbf{Y}$.

Secure Multiplexer. The secure multiplexer protocol Π_{MUX} [25] takes an arithmetic share $\langle x\rangle^{\ell}$ and a Boolean share $\langle b\rangle^{1}$ as input and returns an arithmetic share $\langle z\rangle^{\ell}$, such that $z = x$ if $b = 1$, and $z = 0$ if $b = 0$.

Secure Comparison. The secure comparison protocol Π_{CMP} [24] takes arithmetic shares $\langle x\rangle^{\ell}$ and $\langle y\rangle^{\ell}$ as input and returns a bit share $\langle b\rangle^{1}$, such that $b = 1$ if $x \geq y$, and $b = 0$ otherwise.

Secure Truncation. The secure truncation protocol Π_{Trunc} [12] takes an arithmetic share $\langle x\rangle^{\ell}$ and an integer f as input and returns $\langle z\rangle^{\ell}$, such that $z = x \gg f$.

Truncate and Reduce. Similar to Π_{Trunc}, the secure truncate and reduce protocol Π_{TR} takes $\langle x\rangle^{\ell}$ and f as input, but returns $\langle z\rangle^{\ell-f}$, such that $z = x \gg f$.

Secure Extension. The secure extension Π_{Extend} takes an arithmetic share $\langle x \rangle^{\ell'}$ and a target bit-width ℓ ($> \ell'$) as input and returns $\langle x \rangle^{\ell}$, such that extending x to a larger bit-width.

Secure Top-K. The secure top-K protocol Π_{Topk} [13] takes a vector share $\langle \mathbf{x} \rangle$ and an integer k as input and returns $\langle \mathbf{z} \rangle$, such that $\mathbf{z} = \text{Top-}K(\mathbf{x}, k)$.

Secure Softmax. The secure softmax protocol Π_{Softmax} [21] takes an arithmetic share $\langle x \rangle$ as input and returns $\langle z \rangle$, such that $z = \texttt{Softmax}(x)$.

Secure Lookup Table (LUT). The secure LUT protocol Π_{LUT} [6] takes a public table T and an index share $\langle i \rangle$ as input and returns $\langle z \rangle$, such that $z = T[i]$, i.e. the i-th entry of a public table T with M entries.

B Linear Function Parameters

Table 6 lists the parameters of linear functions used for the `SiLU` approximation. These parameters are fitted using numpy.polyfit API [5].

Table 6: LUT-based `SiLU` linear approximation parameters.

Interval	Slope	Intercept	Interval	Slope	Intercept
$[-8,-7)$	-0.0036530436	-0.0316713527	$[0,1)$	0.7345586419	-0.0378005020
$[-7,-6)$	-0.0083718523	-0.0644570142	$[1,2)$	1.0348082781	-0.3192403913
$[-6,-5)$	-0.0184673574	-0.1245436966	$[2,3)$	1.0974453688	-0.4335394502
$[-5,-4)$	-0.0382546298	-0.2226767540	$[3,4)$	1.0703147650	-0.3502155542
$[-4,-3)$	-0.0703148022	-0.3502154350	$[4,5)$	1.0382546186	-0.2226768285
$[-3,-2)$	-0.0974453092	-0.4335395098	$[5,6)$	1.0184673071	-0.1245449260
$[-2,-1)$	-0.0348083526	-0.3192403316	$[6,7)$	1.0083717108	-0.0644548759
$[-1,0)$	0.2654413581	-0.0378005207	$[7,8)$	1.0036530495	-0.0316730142

C Communication Complexity Analysis

The communication complexity of our proposed secure routing protocol Π_{Routing} is

$$\mathcal{O}\big(Nm + Nm(1+q)\log^2 m + Ntm + Nmd\big), \qquad q = \lceil m/64 \rceil,$$

where q is the number of packed 64-bit words per one-hot token label. The $\mathcal{O}(Nm)$ term accounts for secure softmax. The sorting term arises from N expert-wise odd-even merge sorting networks, each requiring $\mathcal{O}(m\log^2 m)$ compare-and-swap operations that carry a routing score and q Boolean label words. The remaining terms account for Boolean-to-arithmetic conversion of the selected one-hot matrices and secure matrix multiplication for token retrieval, respectively. Public label initialization and local unpacking of the selected Boolean shares require no communication. For the evaluated routing configurations, $m \le 64$ and $t \le d$; therefore, this bound simplifies to $\mathcal{O}(Nm + Nm\log^2 m + Nmd)$.

In contrast, under its Top-k protocol, the communication complexity of `CryptoMoE` is:

$$\mathcal{O}(Nm + Nkm\log(km) + Nmd + mN\log N + Nkm + Ntm).$$

The additional $\mathcal{O}(mN\log N)$ term arises from token-wise expert selection in the first stage. The $\mathcal{O}(Nkm)$ term accounts for secure equality tests and multiplexer operations used for candidate regrouping, while $\mathcal{O}(Ntm)$ accounts for equality-based one-hot selection matrix generation. Capricorn eliminates these token-wise selection and equality-based stages. Its expert-wise selection operates on m scores rather than km candidates, with the communication for carrying packed labels explicitly included in the sorting term above.

[5] `https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html`

D Additional Experimental Results

D.1 Detailed Accuracy Results

Tables 7 and 8 report per-dataset accuracy for the six routing-activation combinations on three MoE models, using batch sizes 16 and 64, respectively. All approximate-`SiLU` configurations are evaluated under fixed-point emulation. Subscripts denote $\pm$ the per-dataset standard error.

Table 7: Per-dataset accuracy for batch size 16. Poly and PWL denote the polynomial and piecewise-linear `SiLU` approximations, respectively.

Model	Routing	SiLU	OBQA	ARC-E	ARC-C	HellaSwag	PIQA	WinoGrande
DeepSeekMoE	Two-stage	Exact	$41.20\%_{\pm 2.20\%}$	$71.93\%_{\pm 0.92\%}$	$46.67\%_{\pm 1.46\%}$	$76.23\%_{\pm 0.42\%}$	$79.92\%_{\pm 0.93\%}$	$69.93\%_{\pm 1.29\%}$
		Poly	$42.60\%_{\pm 2.21\%}$	$71.89\%_{\pm 0.92\%}$	$46.84\%_{\pm 1.46\%}$	$76.30\%_{\pm 0.42\%}$	$80.14\%_{\pm 0.93\%}$	$69.14\%_{\pm 1.30\%}$
		PWL	$41.60\%_{\pm 2.21\%}$	$72.26\%_{\pm 0.92\%}$	$45.73\%_{\pm 1.46\%}$	$75.86\%_{\pm 0.43\%}$	$80.09\%_{\pm 0.93\%}$	$70.24\%_{\pm 1.28\%}$
	One-stage	Exact	$42.80\%_{\pm 2.21\%}$	$73.02\%_{\pm 0.91\%}$	$46.67\%_{\pm 1.46\%}$	$76.22\%_{\pm 0.42\%}$	$79.82\%_{\pm 0.94\%}$	$70.32\%_{\pm 1.28\%}$
		Poly	$42.60\%_{\pm 2.21\%}$	$72.39\%_{\pm 0.92\%}$	$46.42\%_{\pm 1.46\%}$	$76.12\%_{\pm 0.43\%}$	$79.87\%_{\pm 0.94\%}$	$69.38\%_{\pm 1.30\%}$
		PWL	$42.20\%_{\pm 2.21\%}$	$73.27\%_{\pm 0.91\%}$	$46.42\%_{\pm 1.46\%}$	$75.90\%_{\pm 0.43\%}$	$79.82\%_{\pm 0.94\%}$	$70.64\%_{\pm 1.28\%}$
QwenMoE	Two-stage	Exact	$43.60\%_{\pm 2.22\%}$	$69.07\%_{\pm 0.95\%}$	$43.43\%_{\pm 1.45\%}$	$76.73\%_{\pm 0.42\%}$	$79.92\%_{\pm 0.93\%}$	$68.90\%_{\pm 1.30\%}$
		Poly	$43.40\%_{\pm 2.22\%}$	$68.43\%_{\pm 0.95\%}$	$44.80\%_{\pm 1.45\%}$	$76.81\%_{\pm 0.42\%}$	$79.87\%_{\pm 0.94\%}$	$69.30\%_{\pm 1.30\%}$
		PWL	$42.80\%_{\pm 2.21\%}$	$68.52\%_{\pm 0.95\%}$	$44.20\%_{\pm 1.45\%}$	$76.77\%_{\pm 0.42\%}$	$79.76\%_{\pm 0.94\%}$	$69.38\%_{\pm 1.30\%}$
	One-stage	Exact	$43.40\%_{\pm 2.22\%}$	$69.23\%_{\pm 0.95\%}$	$44.28\%_{\pm 1.45\%}$	$76.79\%_{\pm 0.42\%}$	$80.09\%_{\pm 0.93\%}$	$68.98\%_{\pm 1.30\%}$
		Poly	$43.00\%_{\pm 2.22\%}$	$69.44\%_{\pm 0.95\%}$	$44.71\%_{\pm 1.45\%}$	$76.81\%_{\pm 0.42\%}$	$80.36\%_{\pm 0.93\%}$	$70.17\%_{\pm 1.29\%}$
		PWL	$44.40\%_{\pm 2.22\%}$	$69.40\%_{\pm 0.95\%}$	$44.80\%_{\pm 1.45\%}$	$76.72\%_{\pm 0.42\%}$	$80.20\%_{\pm 0.93\%}$	$70.40\%_{\pm 1.28\%}$
OLMoE	Two-stage	Exact	$45.40\%_{\pm 2.23\%}$	$74.83\%_{\pm 0.89\%}$	$48.81\%_{\pm 1.46\%}$	$79.43\%_{\pm 0.40\%}$	$77.20\%_{\pm 0.98\%}$	$67.80\%_{\pm 1.31\%}$
		Poly	$45.40\%_{\pm 2.23\%}$	$74.96\%_{\pm 0.89\%}$	$49.91\%_{\pm 1.46\%}$	$79.50\%_{\pm 0.40\%}$	$76.66\%_{\pm 0.99\%}$	$66.61\%_{\pm 1.33\%}$
		PWL	$43.80\%_{\pm 2.22\%}$	$73.95\%_{\pm 0.90\%}$	$49.23\%_{\pm 1.46\%}$	$79.18\%_{\pm 0.41\%}$	$76.88\%_{\pm 0.98\%}$	$67.09\%_{\pm 1.32\%}$
	One-stage	Exact	$45.20\%_{\pm 2.23\%}$	$75.80\%_{\pm 0.88\%}$	$49.23\%_{\pm 1.46\%}$	$79.45\%_{\pm 0.40\%}$	$77.15\%_{\pm 0.98\%}$	$68.35\%_{\pm 1.31\%}$
		Poly	$45.40\%_{\pm 2.23\%}$	$75.55\%_{\pm 0.88\%}$	$50.60\%_{\pm 1.46\%}$	$79.54\%_{\pm 0.40\%}$	$76.77\%_{\pm 0.99\%}$	$68.43\%_{\pm 1.31\%}$
		PWL	$46.20\%_{\pm 2.23\%}$	$74.75\%_{\pm 0.89\%}$	$49.15\%_{\pm 1.46\%}$	$79.30\%_{\pm 0.40\%}$	$77.58\%_{\pm 0.97\%}$	$69.38\%_{\pm 1.30\%}$

Table 8: Per-dataset accuracy for batch size 64. Poly and PWL denote the polynomial and piecewise-linear `SiLU` approximations, respectively.

Model	Routing	SiLU	OBQA	ARC-E	ARC-C	HellaSwag	PIQA	WinoGrande
DeepSeekMoE	Two-stage	Exact	$42.20\%_{\pm 2.21\%}$	$71.89\%_{\pm 0.92\%}$	$46.59\%_{\pm 1.46\%}$	$76.50\%_{\pm 0.42\%}$	$80.30\%_{\pm 0.93\%}$	$70.09\%_{\pm 1.29\%}$
		Poly	$42.00\%_{\pm 2.21\%}$	$72.10\%_{\pm 0.92\%}$	$46.76\%_{\pm 1.46\%}$	$76.49\%_{\pm 0.42\%}$	$79.98\%_{\pm 0.93\%}$	$70.40\%_{\pm 1.28\%}$
		PWL	$42.20\%_{\pm 2.21\%}$	$72.10\%_{\pm 0.92\%}$	$45.99\%_{\pm 1.46\%}$	$76.10\%_{\pm 0.43\%}$	$79.76\%_{\pm 0.94\%}$	$71.51\%_{\pm 1.27\%}$
	One-stage	Exact	$42.40\%_{\pm 2.21\%}$	$72.64\%_{\pm 0.91\%}$	$46.84\%_{\pm 1.46\%}$	$76.39\%_{\pm 0.42\%}$	$79.92\%_{\pm 0.93\%}$	$70.88\%_{\pm 1.28\%}$
		Poly	$43.00\%_{\pm 2.22\%}$	$72.64\%_{\pm 0.91\%}$	$47.27\%_{\pm 1.46\%}$	$76.48\%_{\pm 0.42\%}$	$79.82\%_{\pm 0.94\%}$	$70.88\%_{\pm 1.28\%}$
		PWL	$41.80\%_{\pm 2.21\%}$	$72.47\%_{\pm 0.92\%}$	$46.42\%_{\pm 1.46\%}$	$76.17\%_{\pm 0.43\%}$	$79.76\%_{\pm 0.94\%}$	$71.27\%_{\pm 1.27\%}$
QwenMoE	Two-stage	Exact	$43.00\%_{\pm 2.22\%}$	$69.15\%_{\pm 0.95\%}$	$44.88\%_{\pm 1.45\%}$	$76.82\%_{\pm 0.42\%}$	$79.38\%_{\pm 0.94\%}$	$68.75\%_{\pm 1.30\%}$
		Poly	$43.00\%_{\pm 2.22\%}$	$68.90\%_{\pm 0.95\%}$	$45.73\%_{\pm 1.46\%}$	$76.95\%_{\pm 0.42\%}$	$80.03\%_{\pm 0.93\%}$	$70.01\%_{\pm 1.29\%}$
		PWL	$43.20\%_{\pm 2.22\%}$	$69.19\%_{\pm 0.95\%}$	$45.31\%_{\pm 1.45\%}$	$76.92\%_{\pm 0.42\%}$	$79.98\%_{\pm 0.93\%}$	$70.17\%_{\pm 1.29\%}$
	One-stage	Exact	$43.40\%_{\pm 2.22\%}$	$69.78\%_{\pm 0.94\%}$	$44.88\%_{\pm 1.45\%}$	$76.98\%_{\pm 0.42\%}$	$79.82\%_{\pm 0.94\%}$	$69.46\%_{\pm 1.29\%}$
		Poly	$43.20\%_{\pm 2.22\%}$	$69.74\%_{\pm 0.94\%}$	$45.48\%_{\pm 1.46\%}$	$77.00\%_{\pm 0.42\%}$	$80.09\%_{\pm 0.93\%}$	$69.93\%_{\pm 1.29\%}$
		PWL	$44.20\%_{\pm 2.22\%}$	$69.78\%_{\pm 0.94\%}$	$45.31\%_{\pm 1.45\%}$	$77.07\%_{\pm 0.42\%}$	$79.98\%_{\pm 0.93\%}$	$70.09\%_{\pm 1.29\%}$
OLMoE	Two-stage	Exact	$44.80\%_{\pm 2.23\%}$	$74.83\%_{\pm 0.89\%}$	$50.00\%_{\pm 1.46\%}$	$79.72\%_{\pm 0.40\%}$	$77.53\%_{\pm 0.97\%}$	$67.56\%_{\pm 1.32\%}$
		Poly	$45.80\%_{\pm 2.23\%}$	$75.72\%_{\pm 0.88\%}$	$49.15\%_{\pm 1.46\%}$	$79.86\%_{\pm 0.40\%}$	$77.20\%_{\pm 0.98\%}$	$67.80\%_{\pm 1.31\%}$
		PWL	$45.20\%_{\pm 2.23\%}$	$74.87\%_{\pm 0.89\%}$	$49.40\%_{\pm 1.46\%}$	$79.65\%_{\pm 0.40\%}$	$77.75\%_{\pm 0.97\%}$	$68.03\%_{\pm 1.31\%}$
	One-stage	Exact	$46.60\%_{\pm 2.23\%}$	$75.13\%_{\pm 0.89\%}$	$49.15\%_{\pm 1.46\%}$	$79.45\%_{\pm 0.40\%}$	$77.37\%_{\pm 0.98\%}$	$68.51\%_{\pm 1.31\%}$
		Poly	$47.00\%_{\pm 2.23\%}$	$76.30\%_{\pm 0.87\%}$	$49.57\%_{\pm 1.46\%}$	$79.75\%_{\pm 0.40\%}$	$77.75\%_{\pm 0.97\%}$	$68.75\%_{\pm 1.30\%}$
		PWL	$45.60\%_{\pm 2.23\%}$	$74.71\%_{\pm 0.89\%}$	$49.57\%_{\pm 1.46\%}$	$79.41\%_{\pm 0.40\%}$	$77.58\%_{\pm 0.97\%}$	$68.67\%_{\pm 1.30\%}$

D.2 End-to-End Evaluation

We conduct an additional complete two-party secure inference experiment on Google V-MoE (`vmoe_s32_last2_ilsvrc2012_randaug_light1`), which contains eight layers, including two MoE layers and six dense layers, with eight experts per MoE layer. Both `Capricorn` and `CryptoMoE` reuse BumbleBee [19] and SPU [21] for the common Transformer backbone and differ only in the MoE-specific protocols. Therefore, the performance difference reflects the effect of their MoE-layer protocols.

As shown in Table 9, `Capricorn` reduces communication by 22.14% relative to `CryptoMoE`. Compared with `CryptoMoE` $_{\text{serial}}$, `Capricorn` reduces latency by 9.74% under LAN and 31.73% under WAN; compared with `CryptoMoE` $_{\text{parallel}}$, the corresponding reductions are 9.97% and 29.68%,

respectively. The end-to-end speedup is less pronounced because only two of V-MoE's eight Transformer blocks contain MoE modules. The shared Transformer backbone accounts for a substantial fraction of the total runtime and receives no benefit from our MoE-specific optimizations.

Table 9: Evaluation of V-MoE under LAN and WAN. Latency and communication are reported per token; peak memory is reported per complete inference.

Network	Framework	Latency (s/token)	Communication (MiB/token)	Peak Memory (GB)
LAN	$\texttt{CryptoMoE}_{\text{serial}}$	$3.90_{\pm 0.029}$	$125.90_{\pm 0.001}$	$34.23_{\pm 0.082}$
	$\texttt{CryptoMoE}_{\text{parallel}}$	$3.91_{\pm 0.028}$	$125.90_{\pm 0.001}$	$34.24_{\pm 0.060}$
	Ours	$3.52_{\pm 0.024}$	$98.02_{\pm 0.000}$	$39.61_{\pm 0.044}$
WAN	$\texttt{CryptoMoE}_{\text{serial}}$	$25.65_{\pm 0.163}$	$125.90_{\pm 0.001}$	$34.15_{\pm 0.082}$
	$\texttt{CryptoMoE}_{\text{parallel}}$	$24.90_{\pm 0.068}$	$125.90_{\pm 0.002}$	$34.17_{\pm 0.050}$
	Ours	$17.51_{\pm 0.046}$	$98.02_{\pm 0.001}$	$39.58_{\pm 0.064}$

E Limitation

Despite the proposed optimizations, `Capricorn` is less efficient when processing a small number of tokens. Additionally, in the LAN setting, the linear operations within expert computation become a bottleneck, which could be optimized in future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the second paragraph of the abstract and "Our Contributions" in the introduction present our contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: It is included in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All information needed to reproduce the main experimental results of the paper is included in Section 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will open-source our code in `https://github.com/sls33/Capricorn`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The implementation details and the information about the model and dataset used in this paper are stated in Section 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The experiments are repeated many times and we report the average results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All results in this paper are evaluated on a server equipped with two AMD EPYC 9554 CPUs (128 cores, 3.1GHz) and 1.1 TB RAM, which is also stated in Section 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the potential positive societal impacts of secure MoE inference in Section 1.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original paper or open-source library that produced the code package or dataset.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A].

Justification: In this work, LLM is used only for editing grammar, spelling and word choice. LLM does not impact the core methodology, scientific rigorousness, or originality of the research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.