Whisper: Efficiently Preserving Both Your Prompt and Model Privacy in LLM Fine-Tuning

Anonymous ACL submission

Abstract

We propose Whisper, an efficient Privacy-Preserving Fine-Tuning (PPFT) framework that provides Dual-privacy for cloud-based APIs by leveraging homomorphic encryption to protect both user inputs and fine-tuned model parameters. Whisper redesigns the encrypted fine-tuning pipeline at the architectural level by introducing a backpropagation-free encrypted fine-tuning paradigm, enabling practical deployments on top of existing Privacy-Preserving Inference (PPI) schemes. To further improve efficiency, Whisper optimizes the underlying cryptographic protocols with a group-based packing strategy and comparison with bootstrapping, substantially increasing throughput and reducing execution overhead for fine-tuning workloads. Extensive experiments demonstrate strong privacy guarantees and competitive model performance, achieving a 7.43-38.39× efficiency improvement over state-of-the-art cryptographic approaches.

1 Introduction

Fine-tuning has become a crucial stage for achieving customization and task adaptation in Large Language Models (LLMs) (Houlsby et al., 2019; Hu et al., 2022), serving as a key driver for their deployment in various domain-specific applications. By retraining pre-trained models on downstream task datasets, fine-tuning can significantly enhance model specialization and practical utility. However, when clients send data to the cloud via fine-tuning APIs provided by model providers, the transmission and storage of such data in plaintext may lead to severe privacy leakage risks.

To mitigate these concerns, several studies have explored Privacy-Preserving Fine-Tuning (PPFT) schemes based on Differential Privacy (DP) (Yu et al., 2022; Du et al., 2023), which aim to protect sensitive information during the cloud-based fine-tuning process. While these approaches can prevent attackers from extracting training data characteristics by launching inference attacks on the server side model (Kandpal et al., 2024), they remain ineffective against privacy threats arising from the plaintext storage and training of data on the cloud. According to a recent security report by Wiz Research[1], sensitive information such as client prompts and log files could be directly retrieved from the ClickHouse cloud database used by DeepSeek. Such privacy leakage risks targeting client inputs are at least as harmful as model inference attacks and are often even more direct and damaging, potentially leading to the irreversible exposure of clients' private data.

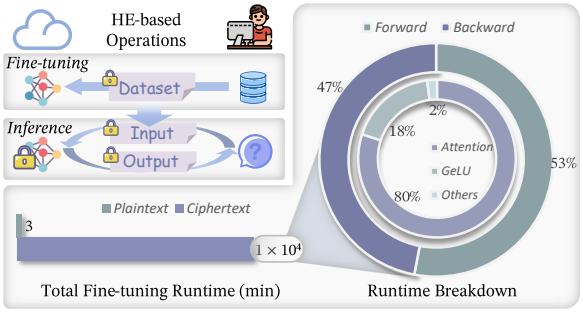


Figure 1: HE-Based Privacy-Preserving Fine-Tuning.

From a cryptographic perspective, privacy-enhancing techniques, such as Homomorphic Encryption (HE) (Cheon et al., 2017) and Secure Multi-Party Computation (MPC) (Micali et al., 1987), can effectively protect the privacy of input data, thereby enabling Privacy-Preserving Machine Learning (PPML) (Cabrero-Holgueras and Pastrana, 2021). One direct application is that clients encrypt their local training data before uploading it, allowing the server to perform parameter updates entirely within the encrypted domain without ever accessing the raw inputs. However, executing forward and backward propagation entirely

[1] https://www.wiz.io/blog/wiz-research-uncovers-exposed-deepseek-database-leak

over encrypted data incurs substantial runtime overhead. As shown in Fig. 1, the HE-based PPFT scheme (Rho et al., 2025) is up to $10^4\times$ slower than plaintext execution, even when fine-tuning a 2-layer BERT$_{base}$ model. The optimized backpropagation still contributes nearly half of the cost, mainly from the Attention and GeLU. Moreover, since fine-tuned models remain on the cloud, all later inference must be performed under encryption, leading to persistent computational overhead.

Consequently, existing PPFT architectures struggle to balance efficiency and privacy. To address this challenge, we propose **Whisper**—an encrypted PPFT framework built on a novel fine-tuning paradigm. Whisper achieves efficient Dual-privacy preservation, simultaneously protecting client inputs and the fine-tuned model while substantially reducing computational overhead:

- **Novel Framework:** Whisper breaks the requirement to perform backpropagation in the encrypted domain by introducing a PPFT framework that relies solely on forward computation. By generating fine-tuned dataset through Privacy-Preserving Inference (PPI), Whisper fully localizes fine-tuning to the client. This ensures that client inputs remain encrypted and that the fine-tuned model parameters stay on the client side, enabling true Dual-privacy.

- **Efficient Protocol:** Whisper builds a non-interactive, high-throughput PPI protocol based on HE, and introduces targeted optimizations for encrypted Attention and GeLU operations. These enhancements enable efficient processing of large-scale concurrent prompts directly in the encrypted domain, thereby allowing rapid generation of the fine-tuned dataset.

- **High Performance:** Whisper enables concurrent multi-prompt PPI for fine-tuned dataset generation, achieving a 7.43-38.39$\times$ speedup over state-of-the-art (SOTA) PPI methods. By introducing no random noise, it improves model accuracy by 4–16% over DP-based approaches and inherently resists Data Reconstruction Attacks through ciphertext-only computation.

2 Preliminaries

2.1 Privacy-Preserving Inference

Privacy-Preserving Inference (PPI) is a central problem in PPML. Since inference involves only forward computation, it is more amenable to encrypted execution, motivating extensive research on secure Transformer inference.

Existing PPI approaches can be broadly categorized into MPC-based and HE-based methods. MPC protocols (Hao et al., 2022; Luo et al., 2024; Pang et al., 2024) rely on interactive execution and incur substantial communication overhead, leading to high latency in Wide-Area Network (WAN) settings. In contrast, HE-based schemes (Chen et al., 2022; Zhang et al., 2025; Park et al., 2025) eliminate interaction by operating directly on ciphertexts, but the computational cost of Transformers is significantly amplified under encryption, resulting in prohibitive runtime overhead. As a result, both paradigms face scalability challenges under high-throughput and high-concurrency workloads.

Beyond cryptographic solutions, several works explore non-cryptographic defenses to improve efficiency. CENTAUR (Luo et al., 2025a) applies vector permutation, while Morris et al. (2023) introduce Gaussian noise at the embedding layer. However, lacking formal security guarantees, these approaches remain vulnerable to Data Reconstruction Attacks (DRA) (Pal et al., 2025), that recover sensitive inputs from intermediate results.

2.2 Privacy-Preserving Fine-Tuning

Privacy-Preserving Fine-Tuning (PPFT) is substantially more challenging than inference, as it requires parameter updates involving backpropagation and gradient computation, which are particularly costly in the encrypted domain. Consequently, existing PPFT research remains limited.

P3EFT (Li et al., 2025) incorporates parameter-efficient techniques such as LoRA into HE-based frameworks to support encrypted fine-tuning on the cloud. However, since the fine-tuned model resides on the server, subsequent inference must still be executed over ciphertexts, incurring persistent computational overhead. Rho et al. (2025) reduce training costs by replacing self-attention with Gaussian kernel attention, yet inference remains fundamentally encrypted.

SecP-Tuning (Luo et al., 2025b) proposes an MPC-based prompt tuning approach for efficient adaptation. However, it optimizes only client-side prompts, which are tightly coupled to a fixed backbone model and become ineffective after model updates, thereby precluding the production of a reusable fine-tuned model for downstream use.

In parallel, DP has been integrated into fine-

tuning to mitigate inference attacks on trained models (Yu et al., 2022; Du et al., 2023). These methods primarily protect model parameters, while the privacy risks associated with the fine-tuning data itself are often insufficiently addressed.

2.3 Homomorphic Encryption

HE enables arbitrary polynomial computations to be performed directly over ciphertexts. In this work, we adopt the CKKS scheme (Cheon et al., 2017), which leverages SIMD encoding (Smart and Vercauteren, 2014) to encrypt a plaintext vector $\boldsymbol{x} \in \mathbb{C}^{N/2}$ into a single ciphertext $\tilde{\boldsymbol{x}} \in \mathcal{R}_q^2$. The scheme supports a variety of algebraic operations in the ciphertext domain, including addition $\widetilde{\boldsymbol{x}+\boldsymbol{y}} \leftarrow \tilde{\boldsymbol{x}} \boxplus \tilde{\boldsymbol{y}}$, constant multiplication $\widetilde{c\boldsymbol{x}} \leftarrow c \boxtimes \tilde{\boldsymbol{x}}$, multiplication $\widetilde{\boldsymbol{x}\boldsymbol{y}} \leftarrow \tilde{\boldsymbol{x}} \boxtimes \tilde{\boldsymbol{y}}$, rotation $\widetilde{\boldsymbol{x}^{\langle r \rangle}} \leftarrow \mathsf{Rot}(\tilde{\boldsymbol{x}}, r)$, bootstrapping $\widetilde{\boldsymbol{x}'} \leftarrow \mathsf{Bts}(\tilde{\boldsymbol{x}})$, and expansion $[\widetilde{\boldsymbol{x}}_0, \widetilde{\boldsymbol{x}}_1, \cdots, \widetilde{\boldsymbol{x}}_{n-1}] \leftarrow \mathsf{Expn}(\tilde{\boldsymbol{x}})$. Formal definitions and implementation details of these operations are provided in the Appendix A.2.

2.4 Transformer

2.4.1 Attention

In Transformer models, Self-Attention (Vaswani et al., 2017) computes attention weights via $\mathsf{Softmax}(\boldsymbol{q}_n \boldsymbol{K}^\top)$ and applies them to the value matrix $\boldsymbol{V}$ to obtain the output for each query vector:

$$\boldsymbol{o}_n = \lambda \boldsymbol{V}, \quad \lambda = \mathsf{Softmax}(\boldsymbol{q}_n \boldsymbol{K}^\top / \sqrt{d}) \tag{1}$$

This computation requires $\mathcal{O}(n)$ inner-product evaluations and an exponential nonlinearity. These operations are particularly costly under encrypted computation and further force HE-based PPML systems to process prompt requests strictly sequentially, thereby limiting overall system throughput.

To mitigate these limitations, recent work has proposed replacing Self-Attention with linear, recurrent-style formulations (Sun et al., 2023; Yang et al., 2024; Sun et al., 2025). These Linear-Attention methods maintain a cumulative state $\boldsymbol{S}_n$ that is updated recursively and produce outputs:

$$\boldsymbol{o}_n = \boldsymbol{S}_n \boldsymbol{q}_n, \quad \boldsymbol{S}_n = D\boldsymbol{S}_{n-1} + \boldsymbol{v}_n \boldsymbol{k}_n^\top \tag{2}$$

Here, D denotes forgetting gate. This formulation removes both the exponential function and the $\mathcal{O}(n)$ inner-product evaluations in Self-Attention, making the effective attention weights depend only on the per-token state $\boldsymbol{S}_n$.(see Appendix A.3 for details). We further find that this property can potentially enable HE-based SIMD techniques to pack and process tokens from multiple prompts in parallel, thereby improving overall system throughput.

2.4.2 GeLU

In Transformer architectures, the GeLU activation introduces a smooth nonlinearity defined via the Gauss error function. Since non-linear functions cannot be directly evaluated in encrypted form, PPML systems typically approximate GeLU using piecewise and high-degree polynomial functions, as formalized in Appendix C. However, HE schemes do not support comparison (Cmp) over ciphertexts, forcing piecewise boundaries to be emulated by additional polynomial approximations. This substantially increases multiplicative depth and accelerates noise growth, often necessitating expensive bootstrapping (Bts) operations (Zhang et al., 2025; Park et al., 2025). As a result, evaluating GeLU becomes one of the dominant computational bottlenecks in HE-based PPML.

3 Whisper

3.1 Encryption-friendly Framework

RQ 1: How can Whisper achieve Encryption-friendly and Dual-privacy PPFT?

Insight 1: Directly implementing PPFT with cryptographic primitives often incurs significant runtime overhead. Section 1 (Fig. 1) shows that current approaches require expensive backpropagation for parameter updates, and the fine-tuned model remains on the server. Consequently, clients are still required to interact with the server in an encrypted manner during subsequent inference.

Inspired by the fine-tuning paradigm recently introduced by OpenAI[2], we observe that a localized fine-tuning mechanism can be achieved through prompt-driven knowledge distillation (Chiang et al., 2023). As illustrated in Fig. 2 (a), the overall Whisper framework proceeds as follows: the client issues prompts $\mathsf{in}^{(i)}, i \in \{\mathsf{prompts}\}$ to a cloud-based teacher model, receives the corresponding distributions $\mathsf{out}^{(i)}$, and then constructs $(\mathsf{in}^{(i)}, \mathsf{out}^{(i)})$ as fine-tuned dataset to locally train its student model.

The key advantage of this framework is that it decouples backpropagation from cloud-side computation, eliminating the high overhead of encrypted gradient updates and reducing PPFT to repeated instances of PPI, which is substantially more efficient under HE than encrypted backpropagation. As a result, Whisper enables direct parameter fine-tuning

[2] https://openai.com/index/api-model-distillation/

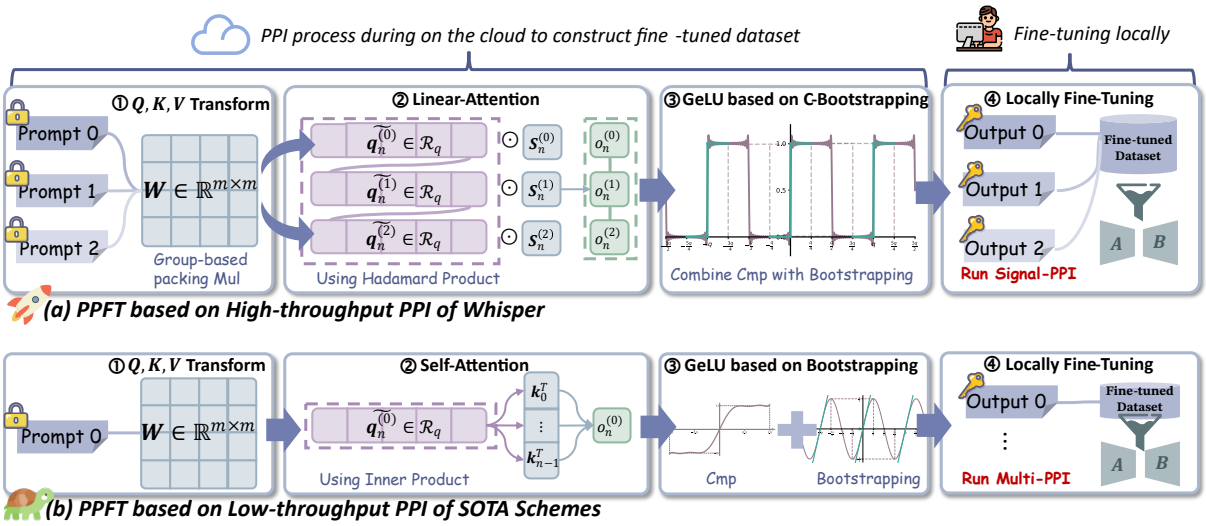


Figure 2: Whisper leverages cloud-side PPI to enable localized fine-tuning: (a) the high-throughput PPI design implemented in Whisper, and (b) the generalized PPI workflow used in current SOTA methods.

of a downstream model entirely on the client, rather than inference-time adaptation or prompt-based tuning. In contrast to vanilla knowledge distillation that relies on public or centrally curated data, Whisper derives task-specific supervision from encrypted prompts via PPI, without ever exposing plaintext data to the teacher.

Threat Model. Under a semi-honest security assumption, Whisper guarantees:

i) *The server learns nothing about client prompts.*

ii) *The client retains exclusive ownership of the fine-tuned model parameters.*

We exclude malicious adversaries and black-box model extraction (Liang et al., 2024), which are inherent to inference-as-a-service and orthogonal to our objectives, without enlarging the attack surface relative to plaintext inference and remaining compatible with standard defenses such as query limiting or model watermarking (Wang et al., 2024).

However, to construct the fine-tuned dataset, the client must issue a large number of prompts and obtain corresponding inference results from the cloud teacher model. This introduces a new technical challenge—how to efficiently support large-scale concurrent prompt inference under encryption.

3.2 Concurrent Processing

RQ 2: How can Whisper efficiently support large-scale concurrent prompt inference under encryption?

Insight 2: Existing PPI schemes based on MPC typically require multiple rounds of interactive communication between the client and the server. Under WAN conditions, such high communication rounds can lead to substantial latency overhead. To mitigate this issue, recent studies have shifted toward non-interactive approaches based on HE, enabling PPI without online interaction.

However, for LLMs, the core Self-Attention mechanism requires computing pairwise similarity scores between tokens within the same prompt. As shown in Step ② of Fig. 2 (b), conventional HE-based implementations pack all tokens of a prompt into a single SIMD ciphertext and rely on homomorphic rotations to compute the required inner products. The resulting attention weights are then obtained via the computationally expensive nonlinear softmax operation, $\lambda \leftarrow \mathsf{Softmax}(q_n K^\top)$, and used to produce the attention output of the n-th token as $o_n = \lambda V$. This Self-Attention-oriented packing strategy restricts the system to processing only single-sample prompt (batch size = 1). As a result, the architecture is forced into a strictly sequential execution pattern, which makes it fundamentally incapable of scaling under high-concurrency inference workloads. Consequently, even SOTA HE-based PPI schemes become impractical when deployed within this PPFT framework.

To address this limitation, we draw inspiration from the Linear-Attention mechanism, which removes the dependency on pairwise token similarities in attention computation. Under this formulation, the attention weight for the n-th token in

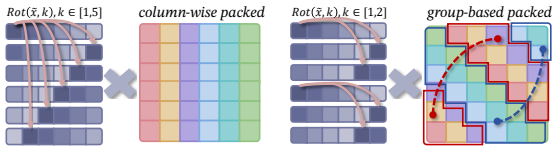
prompt i depends only on its own accumulated state $\boldsymbol{S}_n^{(i)}, i \in \{\text{prompts}\}$ rather than on the similarities among the preceding $n-1$ tokens within the same prompt. As illustrated in Step ② of Fig. 2 (a), this property enables batching tokens from different prompts and packing them into a single SIMD ciphertext. As a result, attention computations can be executed in parallel across prompts, substantially improving inference throughput under high-concurrency workloads.

3.3 Protocol Optimization

RQ 3: How can Whisper improve the overall efficiency of encrypted computation?

Insight 3: Although HE eliminates interaction and reduces communication, ciphertext operations still incur substantial computational overhead—most notably from repeated rotation (key-switching) in linear layers and from the multiplicative depth consumption of polynomial-based non-linear approximations, which together lead to frequent bootstrapping and long execution time. To address these inefficiencies, as shown in Steps ① and ③ of Fig. 2(a), we introduce two targeted optimizations:

3.3.1 Linear Protocol



(a) Naive (b) Whisper ($g = 2$)

Figure 3: Column-wise packing (Naïve) vs. Group-based packing (Whisper).

In Step ① of our framework, the input ciphertext vector $\tilde{\boldsymbol{x}}(\boldsymbol{x} \in \mathbb{C}^{1\times m})$ is linearly mapped to $(\tilde{\boldsymbol{q}}, \tilde{\boldsymbol{k}}, \tilde{\boldsymbol{v}})$ through homomorphic multiplications with plaintext weight matrices $\boldsymbol{W}_{\{Q,K,V\}} \in \mathbb{C}^{m\times m}$. This operation involves plaintext-ciphertext inner product (PCMul) computations, which require multiple homomorphic rotations—a major source of computational overhead. Although prior studies (Juvekar et al., 2018; Pang et al., 2024) have explored optimized packing strategies for the weight matrices to reduce the number of required rotations, the overall efficiency remains constrained by frequent and redundant rotation operations, which are inherently limited by the inefficiency of underlying key-switching procedures.

As illustrated in Fig. 3, we propose a group-based packing scheme built upon the diagonal packing strategy (Halevi and Shoup, 2014). Unlike the naïve approach, which requires an independent rotation for each column vector in the matrix, our method performs diagonal-wise packing of the plaintext weight matrix and further partitions the packed diagonal vectors into g groups, each containing $e = \frac{m}{g}$ vector elements, *i.e.*, $\{\boldsymbol{w}_0^d, \ldots, \boldsymbol{w}_{e-1}^d\}_0, \ldots, \{\boldsymbol{w}_{e(g-1)}^d, \ldots, \boldsymbol{w}_{m-1}^d\}_{g-1}$. This grouping strategy allows different groups to reuse identical rotation operations on the ciphertext $\tilde{\boldsymbol{x}}$, thereby enabling more efficient computation of the inner product as follows:

$$\tilde{z} = \sum_{i=0}^{g-1} \mathsf{Rot}\Big(\sum_{j=0}^{e-1} \mathsf{Rot}(\tilde{\boldsymbol{x}}, j)\boldsymbol{w}_{ie+j}^d, ie\Big) \quad (3)$$

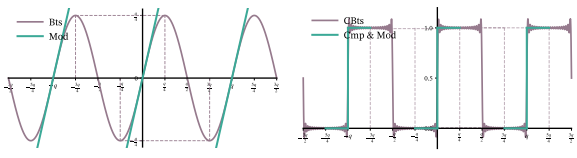
This optimization reduces the rotation complexity from naïve $\mathcal{O}(m^2)$ to $\mathcal{O}(\frac{m}{g} + g - 2)$, which is a convex function with respect to the group size g. Therefore, for a given matrix dimension m, there exists an optimal value of g that minimizes the overall number of rotations.

In practice, performing multiple rotations on the same ciphertext $\tilde{\boldsymbol{x}}$ still incurs repeated key-switching operations, which remain expensive. To further reduce this cost, we adopt the hoisting technique (Halevi and Shoup, 2014), which reuses intermediate ciphertext decomposition results from key-switching to support batch rotation processing:

$$\mathsf{Rot}(\tilde{\boldsymbol{x}}, 1), \ldots, \mathsf{Rot}(\tilde{\boldsymbol{x}}, m-1) \leftarrow \mathsf{Hoist}(\tilde{\boldsymbol{x}}) \quad (4)$$

This optimization allows roughly half of the rotation-related computations to be reused, and integrating it with the group-based packing strategy yields a further substantial performance gain. The detailed linear protocol and the corresponding convex optimization analysis are provided in the Appendix B.

3.3.2 Non-linear Protocol



(a) Bts (b) Whisper (CBts)

Figure 4: Bootstrapping (Naïve) vs. Comparison with Bootstrapping (Whisper).

Since HE schemes only support addition and multiplication over ciphertexts, non-linear functions $f(x)$ are typically approximated by piecewise

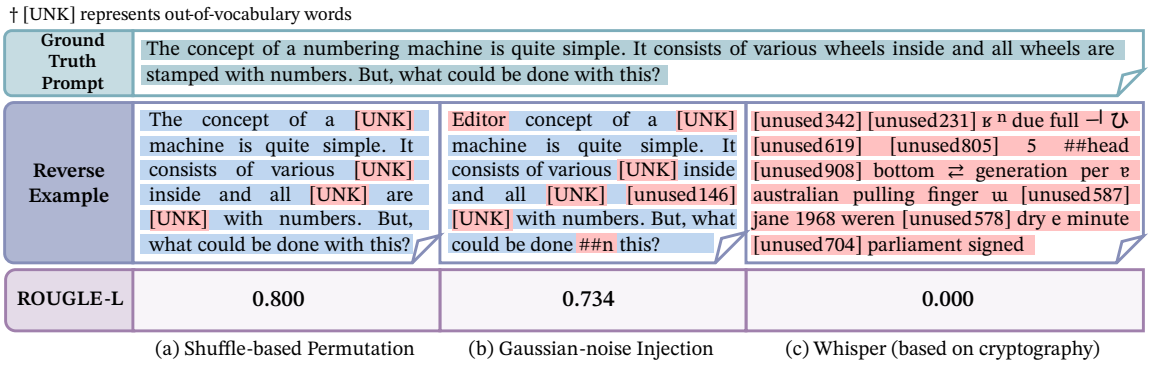


Figure 5: Examples of how different PPI schemes defend against Hidden State Inversion attack.

polynomials, as described in Section 2. However, evaluating such functions often leads to rapid consumption of the modulus budget due to repeated ciphertext multiplications. When the modulus budget is exhausted, a bootstrapping (Bts) operation must be performed to refresh the ciphertext—raising its modulus and restoring its multiplicative depth so that subsequent encrypted computations remain correct. Specifically, given a ciphertext $\tilde{\boldsymbol{x}} \in \mathcal{R}_q^2$, the process first applies ModRaise to lift its modulus to $q' \gg q$, and then performs modular reduction $\tilde{\boldsymbol{x}} + qI \pmod q$ within the interval $[-\frac{q'}{2}, \frac{q'}{2}]$, effectively raising the ciphertext's modular and restoring its multiplicative depth.

However, modular reduction is inherently discontinuous, making it difficult to evaluate directly under HE. Prior work circumvents this by approximating the modulo function with smooth trigonometric forms (Cheon et al., 2018)—for example, replacing it with $(\frac{q}{2\pi})\sin((\frac{2\pi}{q}x))$, as illustrated in Fig. 2 (a). In typical LLM inference pipelines, one Bts operation is already required after the Attention computation. Subsequently, evaluating the piecewise structure of the GeLU approximation introduces an additional non-linear comparison (Cmp) function. Since Cmp must also be approximated by a high-degree polynomial under HE, it consumes substantial multiplicative depth, thereby necessitating yet another Bts to refresh the modulus budget.

Inspired by functional Bts (Alexandru et al., 2025; Yang et al., 2025), we further observe that the approximation polynomial of Cmp can be reformulated with trigonometric components to achieve periodicity. This property enables the Cmp operation to be naturally integrated into the Bts process:

$$\mathsf{Cmp}(x + qI \pmod q) = \frac{1 + \mathsf{Sign}(\sin(\frac{2\pi}{q}x))}{2} \tag{5}$$

This extension enables both Cmp and Bts to be performed within a single homomorphic evaluation, effectively achieving functional fusion and significantly reducing the overall computational cost. We denote this integrated operation as Comparison with Bootstrapping (CBts). Furthermore, we can naturally replace the Sign function with its Fourier sine–series expansion, the expression can be equivalently written as:

$$\frac{1}{2} + \frac{2}{\pi}\sum_{k=1}^{N}\frac{\sin(\frac{2\pi(2k-1)}{q}x)}{2k-1} \tag{6}$$

As illustrated in Fig. 4, the approximation errors introduced by both the polynomial surrogate and the Cmp operation remain consistently small and well controlled, validating the effectiveness of our non-linear function construction. Detailed protocols and analyses for these non-linear components are provided in Appendix C.

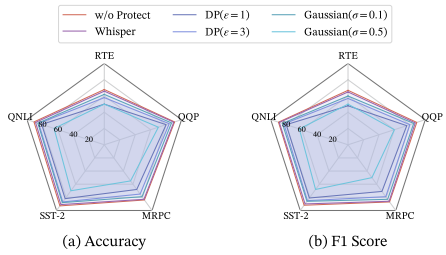
4 Experiments

We evaluate our framework through a three-dimensional assessment that reflects the core properties required for PPFT: **Dimension I (Privacy):** *Does Whisper prevent prompt leakage throughout encrypted pipeline?* **Dimension II (Performance):** *Does Whisper preserve task-level downstream performance?* **Dimension III (Efficiency):** *How much efficiency is gained through our HE optimizations under concurrent workloads?*

Baselines. Since Whisper's framework is compatible with arbitrary PPI schemes for constructing fine-tuned dataset, we include several SOTA cryptographic PPI methods (Pang et al., 2024; Zhang et al., 2025; Park et al., 2025) as baselines to comprehensively evaluate Whisper's efficiency. In addition, we incorporate non-cryptographic

Table 1: Efficiency comparison of PCMul over varying matrix sizes (Runtime in s).

Framework	$\mathbb{C}^{128\times256} \times \mathbb{C}^{256\times256}$				$\mathbb{C}^{256\times256} \times \mathbb{C}^{256\times256}$				$\mathbb{C}^{128\times2048} \times \mathbb{C}^{2048\times2048}$			
	#KS	#Mul	Runtime	Factor	#KS	#Mul	Runtime	Factor	#KS	#Mul	Runtime	Factor
BOLT	136	128	0.19	1.00 ×	264	256	0.47	1.00 ×	3264	128	5.72	1.00 ×
NEXUS	514	32768	0.93	▼4.89 ×	1026	66536	2.16	▼4.60 ×	4112	262144	9.34	▼1.63 ×
Whisper	62	128	0.08	▲2.38 ×	124	256	0.23	▲2.04 ×	1552	128	2.23	▲2.57 ×



(a) Accuracy (b) F1 Score

Figure 6: Accuracy and F1 performance on the GLUE.

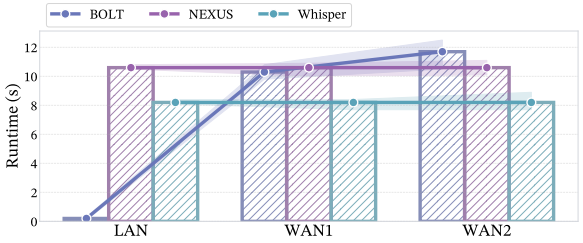


Figure 7: Efficiency Comparison of GeLU on $\mathbb{C}^{2^{15}}$.

PPI schemes, including Shuffle-based permutation method (Luo et al., 2025a) and Gaussian-noise injection at the embedding layer (Morris et al., 2023), as well as DP-based PPFT approaches (Du et al., 2023) that perturb model parameters directly. These baselines enable a thorough comparison of Whisper's advantages in downstream performance and robustness.

Models and Datasets. To ensure comparability with prior work, we conduct experiments on both BERT and GPT-2, and evaluate performance across multiple standard datasets from the GLUE benchmark (Wang et al., 2018) (RTE, QQP, MRPC, SST-2, QNLI) and Fineweb-Edu dataset (Penedo et al., 2024). Additional experimental details are provided in the Appendix D.

4.1 Privacy

We first note that the newly identified Hidden State Inversion (HSI) attack (Pal et al., 2025) provides significantly stronger reconstruction ability than prior DRA methods, enabling highly accurate recovery of token sequences. To evaluate Whisper's prompt-privacy under this stronger threat model, we benchmark it against non-cryptographic defenses, including Gaussian-noise injection and Shuffle-based permutation.

Attack Objective and Metric. Given the intermediate hidden representation at layer l, $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N] \in \mathbb{C}^{N\times d}$. The attacker aims to reconstruct the input sequence $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$. For each position i and candidate token from vocabulary $\boldsymbol{v} \in \mathcal{V}$, a forward pass yields $\hat{\boldsymbol{h}}_i(\boldsymbol{v}) = f_l(\boldsymbol{x}_{1:i-1}, \boldsymbol{v})$, the attacker selects the sorted L_1 distance as a fuzzy-matching criterion to iteratively recover the original sequence. Following (Pal et al., 2025), we adopt the ROUGE-L score to quantify the similarity between the reconstructed text and the ground-truth prompt. This metric captures both word-order and content-level overlap. Scores closer to 1 indicate more accurate reconstruction, and thus higher privacy leakage.

Evaluation. As shown in Fig. 5 on FineWeb-Edu dataset, Shuffle-based and Gaussian-noise defenses still yield high ROUGE-L scores (0.800 and 0.734), indicating that the reconstructed text remains closely aligned with the ground-truth prompt. This shows that plaintext-domain defenses based on noise injection or permutation fail to prevent reconstruction from intermediate representations. In contrast, Whisper achieves the lowest ROUGE-L score, producing semantically meaningless outputs. Without the client-side secret key, ciphertext-token reconstruction is computationally infeasible, highlighting the advantage of cryptographic protection for prompt privacy.

4.2 Performance

We benchmark the downstream performance on GLUE after fine-tuning a BERT_{base} model under four settings: (i) fine-tuning without any privacy protection, (ii) DP-based method, (iii) Gaussian-noise injection method, and (iv) our Whisper-based privacy-preserving approach.

As shown in Fig. 6, Whisper achieves down-

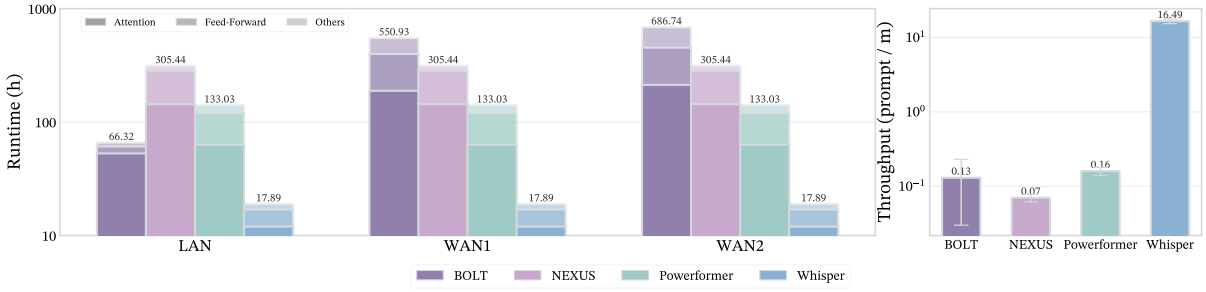


Figure 8: Runtime and throughput for generating a 128-sample fine-tuned dataset under different PPI schemes.

stream fine-tuning performance that is almost indistinguishable from the non-private baseline, with an average accuracy drop of less than 3% across all evaluated tasks. In contrast, training with DP leads to a noticeable degradation, reducing accuracy by 10-18% when privacy budget ϵ is set between 1 and 3. Similarly, directly injecting Gaussian noise into the embedding layer causes consistent performance loss: when the noise standard deviation is $\sigma = 0.1$, accuracy drops by approximately 4-6%, and further degrades by more than 18% when σ increases to 0.5. Unlike noise-based methods that disrupt gradients or representations, Whisper avoids stochastic perturbations and preserves optimization stability through accurate non-linear approximations, enabling strong privacy protection without sacrificing model performance.

4.3 Efficiency

We conduct a comprehensive evaluation of Whisper's efficiency improvements on both linear and non-linear layers, and further benchmark the processing time of several SOTA PPI schemes for constructing fine-tuned dataset under different network conditions (bandwidth, round-trip latency): LAN (3Gbps, 0.8ms), WAN1 (200Mbps, 40ms), and WAN2 (200Mbps, 80ms).

Linear. In the linear-layer evaluation (Table 1), Whisper substantially reduces the number of Rot operation, yielding runtime reductions of 2.04-2.57× and 4.18-11.64× compared to BOLT and NEXUS, respectively. This improvement primarily stems from our proposed group-optimization strategy, which effectively lowers the rotation complexity of inner-product computations. In addition, the incorporation of hoisting techniques enables reuse of key-switch primitives, further decreasing the number of rotations that must actually be executed.

Non-linear. In the non-linear-layer evaluation (Fig. 7), we focus on the efficiency of the GeLU operation. BOLT, which relies on MPC protocols, incurs multiple rounds of communication and thus suffers increasingly high latency under degraded network conditions. Although NEXUS avoids communication due to its fully homomorphic construction, the separation of Cmp and Bts evaluation causes its runtime to exceed that of BOLT in certain network settings. In contrast, Whisper integrates Cmp and Bts into a single operation via CBts, achieving significantly improved efficiency and reducing the runtime by 10.87-24.67% and 20.29% relative to BOLT and NEXUS, respectively.

Dataset Generation. Since fine-tuning is performed in plaintext, dataset generation via PPI becomes the dominant efficiency bottleneck in PPFT. Accordingly, in Fig. 8 we measure the time required by various SOTA PPI schemes to process 128 prompts on BERT_{base} model and generate 128 outputs, each containing 10 tokens, to form a fine-tuned dataset. Whisper achieves substantial acceleration primarily due to its underlying Linear-Attention architecture: by decoupling the pairwise similarity computation among tokens from the accumulated state S, Whisper enables concurrent processing of multiple prompts, leading to a 103.06-235.57× improvement in throughput and a 7.43-38.39× reduction in runtime.

5 Conclusion

We present Whisper, an efficient HE-based PPFT framework that protects both user inputs and fine-tuned model privacy. To improve efficiency, Whisper relies on forward-only PPI, eliminating the need for costly encrypted backpropagation. We further design a suite of optimized protocols for both linear and nonlinear layers in language models. Extensive experiments demonstrate the effectiveness and practicality of Whisper.

Limitations

Following the standard security assumptions in HE-based PPML (Chen et al., 2022; Lee et al., 2023; Zhang et al., 2025; Park et al., 2025), our framework adopts a semi-honest security model, under which all parties are assumed to follow the prescribed protocol execution. While this assumption does not provide protection against fully malicious participants that arbitrarily deviate from the protocol, it remains practical in real-world deployments.

Minimal Server-Side Security Assumption. In particular, although a malicious server could in principle deviate from the prescribed protocol, enforcing stronger guarantees against such behavior would typically require interactive mechanisms, such as MPC protocols augmented with message authentication codes (MACs) (Keller et al., 2016; Escudero et al., 2022) or zero-knowledge proofs (Atapoor et al., 2024; Zhou et al., 2025), to verify correct execution. In contrast, the threat model of Whisper imposes only a minimal security requirement, namely that the server must not be able to access any client plaintext information. Under our HE-based design, inference is entirely non-interactive and all computations are performed directly over encrypted data, which inherently prevents the server from observing or reconstructing any plaintext-sensitive information. As a result, this guarantee suffices for our threat model and remains consistent with the standard assumptions adopted in existing HE-based PPML systems.

Client Incentive Compatibility. Conversely, the client's objective is to fine-tune a local model to improve performance on target downstream tasks. Any deviation from the prescribed protocol would be self-defeating, as it would prevent the client from acquiring useful knowledge from the teacher model and thus undermine this objective. Under this setting, Whisper does not aim to prevent black-box extraction of the teacher model via repeated queries (Tramèr et al., 2016; Liang et al., 2024), which is orthogonal to our goals and can be addressed through standard complementary defenses such as query rate limiting or model watermarking (Jia et al., 2021; Wang et al., 2024).

Therefore, the adopted security model strikes a pragmatic balance between efficiency and protection, providing sufficient guarantees for practical deployment, while not enlarging the attack surface compared to plaintext inference.

References

Andreea Alexandru, Andrey Kim, and Yuriy Polyakov. 2025. General functional bootstrapping using ckks. In *CRYPTO 2025*, pages 304–337. Springer.

Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. 2018. Pir with compressed queries and amortized query processing. In *2018 IEEE symposium on security and privacy (IEEE S&P 2018)*, pages 962–979. IEEE.

Shahla Atapoor, Karim Baghery, Hilder VL Pereira, and Jannik Spiessens. 2024. Verifiable fhe via lattice-based snarks. *IACR Communications in Cryptology (CiC 2024)*, 1(1).

José Cabrero-Holgueras and Sergio Pastrana. 2021. Sok: Privacy-preserving computation techniques for deep learning. *Proceedings on Privacy Enhancing Technologies (PETs 2021)*.

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 of the Association for Computational Linguistics (ACL 2022)*, pages 3510–3520.

Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. 2018. Bootstrapping for approximate homomorphic encryption. In *EUROCRYPT 2018*, pages 360–384. Springer.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT 2017*, pages 409–437. Springer.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Minxin Du, Xiang Yue, Sherman SM Chow, Tianhao Wang, Chenyu Huang, and Huan Sun. 2023. Dp-forward: Fine-tuning and inference on language models with differential privacy in forward pass. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS 2023)*, pages 2665–2679.

Daniel Escudero, Chaoping Xing, and Chen Yuan. 2022. More efficient dishonest majority secure computation over $\mathbb{Z}_{2^k}$ via galois rings. In *CRYPTO 2022*, pages 383–412. Springer.

Shai Halevi and Victor Shoup. 2014. Algorithms in helib. In *CRYPTO 2014*, pages 554–571. Springer.

Shai Halevi and Victor Shoup. 2021. Bootstrapping for helib. *Journal of Cryptology*, 34(1):7.

Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. 2022. Iron: Private inference on transformers. *Advances in Neural Information Irocessing Systems (NeurIPS 2022)*, 35:15718–15731.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning (ICML 2019)*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations (ICLR 2022)*, 1(2):3.

Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. 2021. Entangled watermarks as a defense against model extraction. In *30th USENIX Security Symposium (USENIX Security 2021)*, pages 1937–1954.

Charanjit S Jutla and Nathan Manohar. 2022. Sine series approximation of the mod function for bootstrapping of approximate he. In *EUROCRYPT 2022*, pages 491–520. Springer.

Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. {GAZELLE}: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 2018)*, pages 1651–1669.

Nikhil Kandpal, Krishna Pillutla, Alina Oprea, Peter Kairouz, Christopher A Choquette-Choo, and Zheng Xu. 2024. User inference attacks on large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP 2024)*, pages 18238–18265.

Marcel Keller, Emmanuela Orsini, and Peter Scholl. 2016. Mascot: faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS 2016)*, pages 830–842.

Seewoo Lee, Garam Lee, Jung Woo Kim, Junbum Shin, and Mun-Kyu Lee. 2023. Hetal: Efficient privacy-preserving transfer learning with homomorphic encryption. In *International Conference on Machine Learning (ICML 2023)*, pages 19010–19035. PMLR.

Yang Li, Wenhan Yu, and Jun Zhao. 2025. Privtuner with homomorphic encryption and lora: A p3eft scheme for privacy-preserving parameter-efficient fine-tuning of ai foundation models. *IEEE Transactions on Wireless Communications*.

Jiacheng Liang, Ren Pang, Changjiang Li, and Ting Wang. 2024. Model extraction attacks revisited. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security (AsiaCCS 2024)*, pages 1231–1245.

Jinglong Luo, Guanzhong Chen, Yehong Zhang, Shiyu Liu, Hui Wang, Yue Yu, Xun Zhou, Yuan Qi, and Zenglin Xu. 2025a. Centaur: Bridging the impossible trinity of privacy, efficiency, and performance in privacy-preserving transformer inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, pages 22751–22770.

Jinglong Luo, Yehong Zhang, Zhuo Zhang, Jiaqi Zhang, Xin Mu, Hui Wang, Yue Yu, and Zenglin Xu. 2024. 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.

Jinglong Luo, Zhuo Zhang, Yehong Zhang, Shiyu Liu, Ye Dong, Hui Wang, Yue Yu, Xun Zhou, and Zenglin Xu. 2025b. Secp-tuning: Efficient privacy-preserving prompt tuning for large language models via mpc. *arXiv preprint arXiv:2506.15307*.

Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*, pages 1–23. Springer.

Silvio Micali, Oded Goldreich, and Avi Wigderson. 1987. How to play any mental game. In *Proceedings of the Nineteenth ACM Symp. on Theory of Computing (STOC 1987)*, pages 218–229. ACM New York.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M Rush. 2023. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*, pages 12448–12460.

Arka Pal, Rahul Krishna Thomas, Louai Zahran, Erica Choi, Akilesh Potti, and Micah Goldblum. 2025. Hidden no more: Attacking and defending private third-party llm inference. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. 2024. Bolt: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy (IEEE S&P 2024)*, pages 4753–4771. IEEE.

Dongjin Park, Eunsang Lee, and Joon-Woo Lee. 2025. Powerformer: Efficient and high-accuracy privacy-preserving language model with homomorphic encryption. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, pages 11090–11111.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems (NeruIPS 2024)*, 37:30811–30849.

Donghwan Rho, Taeseong Kim, Minje Park, Jung Woo Kim, Hyunsik Chae, Ernest K Ryu, and Jung Hee Cheon. 2025. Encryption-friendly llm architecture. In *International Conference on Learning Representations (ICLR 2025)*.

Nigel P Smart and Frederik Vercauteren. 2014. Fully homomorphic simd operations. *Designs, codes and cryptography*, 71(1):57–81.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, and 1 others. 2025. Learning to (learn at test time): Rnns with expressive hidden states. In *International Conference on Machine Learning (ICML 2025)*.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction {APIs}. In *25th USENIX Security Symposium (USENIX Security 2016)*, pages 601–618.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS 2017)*, 30.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355.

Zhenyi Wang, Yihan Wu, and Heng Huang. 2024. Defense against model extraction attack by bayesian active watermarking. In *International Conference on Machine Learning (ICML 2024)*.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. 2024. Parallelizing linear transformers with the delta rule over sequence length. *Advances in Neural Information Processing Systems (NeurIPS 2024)*, 37:115491–115522.

Zhaomin Yang, Chao Niu, Benqiang Wei, Zhicong Huang, Cheng Hong, and Tao Wei. 2025. Rboot: Accelerating homomorphic neural network inference by fusing relu within bootstrapping. *Cryptology ePrint Archive*.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, and 1 others. 2022. Differentially private fine-tuning of language models. In *International Conference on Learning Representations (ICLR 2022)*, pages 1–19.

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

Zhelei Zhou, Yun Li, Yuchen Wang, Zhaomin Yang, Bingsheng Zhang, Cheng Hong, Tao Wei, and Wenguang Chen. 2025. Zhe: Efficient zero-knowledge proofs for he evaluations. In *2025 IEEE Symposium on Security and Privacy (IEEE S&P 2025)*, pages 3328–3346. IEEE.

A Details of Preliminaries

A.1 Notation

The notation used in this paper is summarized in Table 2.

Table 2: Summary of notations.

Notation	Description
$x, \boldsymbol{x}, \boldsymbol{X}$	Scalar, vector, and matrix, respectively.
$\tilde{\boldsymbol{x}}$	HE ciphertext vector.
$\tilde{\mathbf{x}}$	Batched HE ciphertext vector.
χ	Discrete gaussian distribution.
ℓ	Multiplicative depth.
N	Polynomial degree in the HE scheme.

A.2 Homomorphic Encryption

Homomorphic Encryption (HE) supports arbitrary polynomial operations over ciphertexts. In this work, we adopt the CKKS scheme (Cheon et al., 2017), which enables mapping plaintext complex vectors in $\mathbb{C}^{N/2}$ into ciphertext polynomials over the ring $\mathcal{R}_q \leftarrow \mathbb{Z}[X]/(X^N+1)$ for approximate homomorphic computation. The ciphertext modulus q is organized as a modulus chain $q_\ell \leftarrow q_0 \cdot \Pi_{i=1}^{\ell} Q_i$, allowing up to ℓ multiplicative levels. The homomorphic operations used in this paper are summarized as follows.

- **KeyGen:** CKKS employs the RLWE assumption (Lyubashevsky et al., 2010) to generate a key pair $(\mathsf{sk}, \mathsf{pk})$. The secret key is sampled as $s \leftarrow \chi$ in $\mathcal{R}_q$. The public key is defined as $\mathsf{pk} = (b, a) \in \mathcal{R}_q^2$, where $b = -as + e, a \leftarrow \mathbb{Z}_q, e \leftarrow \chi$.

- **SIMD Encoding:** A complex vector $\boldsymbol{x} \in \mathbb{C}^{N/2}$ is packed into the $N/2$ SIMD slots of a polynomial and scaled by a fixed-point factor Δ, producing an encoded plaintext $\boldsymbol{m} \leftarrow \mathsf{Encode}(\boldsymbol{x})$, represented as an integer polynomial $\boldsymbol{m}_\Delta = \Delta \cdot \boldsymbol{m} \in \mathcal{R}$.

- **Encryption:** Given a plaintext polynomial $\boldsymbol{m}_\Delta$, the ciphertext is generated as $\tilde{\boldsymbol{x}} = (c_0, c_1) = (bu + e_1 + \boldsymbol{m}_\Delta, au + e_2)$, where $u, e_1, e_2 \leftarrow \chi$.

- **Decryption:** Using the secret key, the plaintext is recovered as $\boldsymbol{m}_\Delta = c_0 + c_1 s \pmod q$, followed by decoding and rescaling.

- **Addition:** For ciphertexts $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$, homomorphic addition is defined as $\widetilde{\boldsymbol{x}+\boldsymbol{y}} \leftarrow \tilde{\boldsymbol{x}} \boxplus \tilde{\boldsymbol{y}}$.

- **Multiplication:** For ciphertexts $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$, homomorphic multiplication is defined as $\widetilde{\boldsymbol{x} \times \boldsymbol{y}} \leftarrow \tilde{\boldsymbol{x}} \boxtimes \tilde{\boldsymbol{y}}$, followed by relinearization and rescaling as required by CKKS.

- **Rotation:** A ciphertext $\tilde{\boldsymbol{x}}$ can be rotated by r SIMD slots using the rotation operation $\widetilde{\boldsymbol{x}^{\langle r \rangle}} \leftarrow \mathsf{Rot}(\tilde{\boldsymbol{x}}, r)$.

- **Expansion:** Given a ciphertext $\tilde{\boldsymbol{x}}$, the expansion operator produces a vector of ciphertexts $[\tilde{\boldsymbol{x}}_0, \tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_{m-1}] \leftarrow \mathsf{Expn}(\tilde{\boldsymbol{x}})$, with technical details provided in (Angel et al., 2018).

- **Bootstrapping:** Given a ciphertext $\tilde{\boldsymbol{x}} \in \mathcal{R}_q^2$, the bootstrapping operation lifts it to a larger modulus $q' \gg q$, thereby restoring noise budget and enabling additional multiplicative levels: $\tilde{\boldsymbol{x}'} \in \mathcal{R}_{q'}^2 \leftarrow \mathsf{Bts}(\tilde{\boldsymbol{x}})$. Further details are provided (Halevi and Shoup, 2021).

A.3 Transformer

The Transformer, exemplified by models such as BERT and GPT, is a deep neural architecture composed of a stack of identical blocks, each containing an Attention module followed by a Feed-Forward Network (FFN). In this work, we focus solely on the computations involved in the inference stage.

Self-Attention. Given an input matrix $\boldsymbol{X}$, the model first projects it into the corresponding query, key, and value representations:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q, \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K, \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V \tag{7}$$

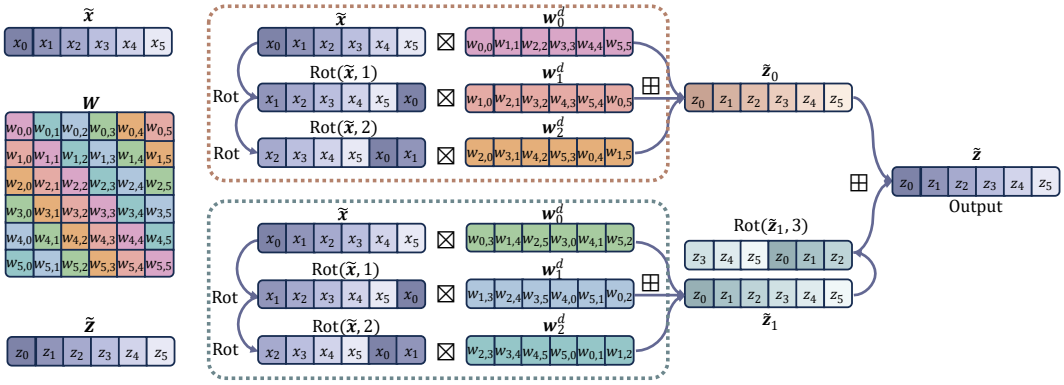
For query vector $\boldsymbol{q}_n$ of token n, Self-Attention is computed as:

$$\boldsymbol{o}_n = \lambda \boldsymbol{V}, \quad \lambda = \mathsf{Softmax}(\boldsymbol{q}_n \boldsymbol{K}^\top / \sqrt{d_k}) \tag{8}$$

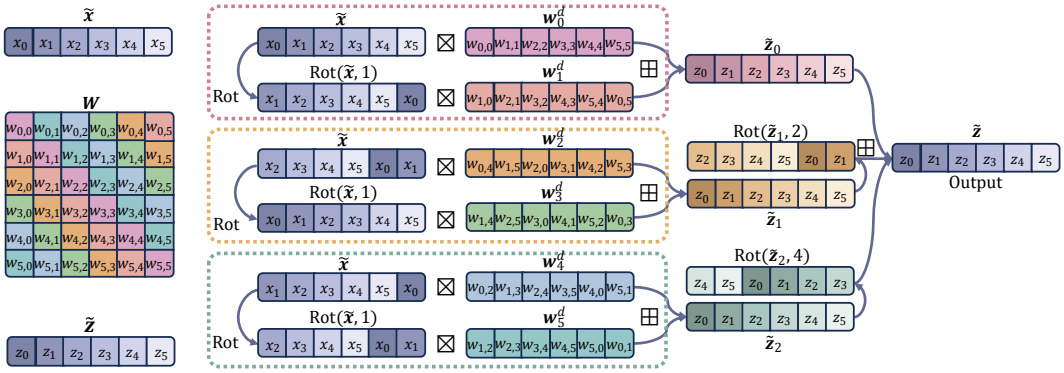
This mechanism captures contextual semantic relationships by measuring the similarity between $\boldsymbol{q}_n$ and all key vectors through inner-product scores, followed by $\mathsf{Softmax}$ normalization to obtain attention score. However, this process inevitably incurs $\mathcal{O}(n)$ inner-product computations per token, along with the additional cost of $\mathsf{Softmax}$ normalization.

Linear-Attention. To reduce the computational complexity of Self-Attention, several works introduce recurrent-style formulations into the attention mechanism, leading to the class of Linear-Attention methods:

$$\boldsymbol{o}_n = \boldsymbol{S}_n \boldsymbol{q}_n, \quad \boldsymbol{S}_n = D\boldsymbol{S}_{n-1} + \boldsymbol{v}_n \boldsymbol{k}_n^\top \tag{9}$$



(a) group-based packing with the number of groups $g = 2$.



(b) group-based packing with the number of groups $g = 3$.

Figure 9: PCMul implemented with the group-based packing strategy, with 9a and 9b corresponding to different group sizes. The illustrated example uses $\boldsymbol{z} = \boldsymbol{x} \times \boldsymbol{W}$, where $\boldsymbol{x} \in \mathbb{C}^{1\times 6}$, $\boldsymbol{W} \in \mathbb{C}^{6\times 6}$, and $\boldsymbol{z} \in \mathbb{C}^{1\times 6}$.

By maintaining an accumulated state $\boldsymbol{S}$, the model captures the semantic relationship between the current token n and all previous tokens, and uses this as the attention score. This formulation effectively removes the $\mathcal{O}(n)$ inner-product operations and the Softmax normalization required in standard Self-Attention.

The state update depends on incorporating the current token information $(\boldsymbol{k}_n, \boldsymbol{v}_n)$, while forgetting outdated token contributions. Existing schemes differ in how they design the forgetting gate D. RetNet (Sun et al., 2023) applies a decay factor γ to gradually forget older token information. DeltaNet (Yang et al., 2024) updates the state by subtracting $(\boldsymbol{S}_{n-1}\boldsymbol{k}_n)\boldsymbol{k}_n^\top$, thereby removing the component of $\boldsymbol{S}_{n-1}$ that is associated with $\boldsymbol{k}_n$. TTT (Sun et al., 2025) instead constructs an optimizer over the current token information $(\boldsymbol{v}_n, \boldsymbol{k}_n)$ to produce the updated state $\boldsymbol{S}_n$.

GeLU. The Feed-Forward Network (FFN) introduces nonlinearity independently for each token:

$$\mathsf{FFN}(\boldsymbol{x}) = \boldsymbol{W}_2\mathsf{GeLU}(\boldsymbol{W}_1\boldsymbol{x} + b_1) + b_2 \tag{10}$$

Both BERT and GPT employ the Gaussian Error Linear Unit (GeLU) as the activation function, defined as:

$$\begin{aligned}\mathsf{GeLU}(x) &= x \cdot \Phi(x), \\ &\approx 0.5x\left(1 + \tanh(\alpha(x + \beta x^3))\right) \\ &\text{where } \alpha = \sqrt{\tfrac{2}{\pi}},\ \beta = 0.044715.\end{aligned} \tag{11}$$

where $\Phi(\cdot)$ denotes the standard Gaussian cumulative distribution function. In practice, GeLU is often approximated using a smooth tanh-based formulation.

Since HE supports only polynomial operations, existing approaches (Zhang et al., 2025; Park et al., 2025) further approximate the GeLU function using piecewise polynomial representations.

B Details of Linear Protocol

B.1 Plaintext-Ciphertext Multiplication

Group Optimization. Inspired by diagonal packing (Halevi and Shoup, 2014) and the BSGS technique (Halevi and Shoup, 2021), we introduce a group-based packing strategy for plaintext-ciphertext multiplication (PCMul) that substan-

tially reduces the number of required homomorphic rotations.

As illustrated in Fig. 9, we begin by reorganizing the plaintext weight matrix $\boldsymbol{W} \in \mathbb{C}^{m\times m}$ into its diagonal vectors and subsequently partitioning these diagonals into g groups, each containing $e = \lceil \frac{m}{g} \rceil$ weight vectors: $\{\boldsymbol{w}_0^d, \ldots, \boldsymbol{w}_{e-1}^d\}_0, \ldots, \{\boldsymbol{w}_{e(g-1)}^d, \ldots, \boldsymbol{w}_{m-1}^d\}_{g-1}$. For each group $i \in [0, g)$, the ciphertext vector $\tilde{\boldsymbol{x}}(\boldsymbol{x} \in \mathbb{C}^{1\times m})$ requires $e-1$ rotations, *i.e.*, $\mathsf{Rot}(\tilde{\boldsymbol{x}}, j)$ for $j \in [0, e)$. The corresponding plaintext weights $\boldsymbol{w}_j^d$ can be shifted directly in the ciphertext and multiplied with the rotated $\mathsf{Rot}(\tilde{\boldsymbol{x}}, j)$ to obtain intermediate results $\tilde{\boldsymbol{z}}_j$. Since all groups reuse the same rotation outputs, the total number of rotations applied to $\tilde{\boldsymbol{x}}$ remains exactly $e-1$.

We then aggregate the intermediate results within each group:

$$\tilde{\boldsymbol{z}}_i = \sum_{j=1}^{e} \tilde{\boldsymbol{z}}_{ie+j} = \sum_{j=0}^{e-1} \mathsf{Rot}(\tilde{\boldsymbol{x}}, j)\boldsymbol{w}_{ie+j} \tag{12}$$

and further rotate and sum the group-level results to produce the final output:

$$\tilde{\boldsymbol{z}} = \sum_{i=0}^{g-1} \mathsf{Rot}(\tilde{\boldsymbol{z}}_i, ie) \tag{13}$$

In terms of complexity, the ciphertext vector $\tilde{\boldsymbol{x}}$ incurs $e-1$ rotations, while the aggregation across all groups requires an additional $g-1$ rotations, yielding a total rotation cost of $\mathcal{O}(e+g-2)$ along with n plaintext-ciphertext multiplications. By further applying hoisting to reuse key-switch intermediates, the effective rotation cost can be reduced to $\mathcal{O}((e+g-2)/2)$.

Crucially, this grouping mechanism turns the rotation complexity into a convex optimization problem: for any given m, there exists an optimal number of groups g that minimizes the overall cost. This property provides a principled basis for selecting parameters when designing efficient homomorphic linear transformations.

Convexity of the Rotation Complexity. The convex optimization analysis of the rotation complexity of PCMul, enabled by our group-based packing strategy, is presented as follows.

Theorem 1 (Rotation Convexity) *Let the rotation complexity:*

$$f(g) = \frac{m}{g} + g - 2 \tag{14}$$

where $m > 0$ is a fixed constant and $g > 0$. Then $f(g)$ is strictly convex over its domain $(0, \infty)$.

Proof 1 *To establish convexity, we analyze the second-order derivative of $f(g)$.*

Step 1: Compute first and second derivatives. *The first derivative is*

$$f'(g) = 1 - \frac{m}{g^2} \tag{15}$$

Differentiating again, the second derivative becomes

$$f''(g) = \frac{2m}{g^3} \tag{16}$$

Step 2: Verify positivity of the second derivative. *Because the domain satisfies $g > 0$, we have $g^3 > 0$. Since the problem further assumes $m > 0$, it follows that*

$$f''(g) = \frac{2m}{g^3} > 0, \quad \textit{for all } g > 0 \tag{17}$$

Step 3: Conclude convexity. *The strict positivity of the second derivative on the entire domain directly implies that $f(g)$ is strictly convex on $(0, \infty)$.*

Batched Group Optimization. The example above illustrates the computation for a single prompt token. In general, the number of available SIMD slots n in the homomorphic polynomial ring is typically larger than the vector dimension m, naturally enabling ciphertext batching. Leveraging this property, our protocol—shown in Fig. 10—can pack token vectors from multiple prompts into a single ciphertext, thereby supporting cross-prompt batched linear transformations.

Formally, consider tokens from $\tau = \lfloor n/m \rfloor$ prompts, denoted as $[\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau-1)}] \in \mathbb{C}^{\tau\times m}$. We first reorder the elements in column-major order and pack them into a single ciphertext vector:

$$\tilde{\mathbf{x}} \leftarrow \mathsf{Enc}(x_0^{(0)}, \cdots, x_0^{(\tau-1)} || \cdots || x_{m-1}^{(0)}, \cdots, x_{m-1}^{(\tau-1)}) \tag{18}$$

Following the same procedure as in the non-batched setting, the plaintext weight matrix $\boldsymbol{W}$ is reorganized into a series of diagonal-wise vectors and partitioned into g groups. To match the packing

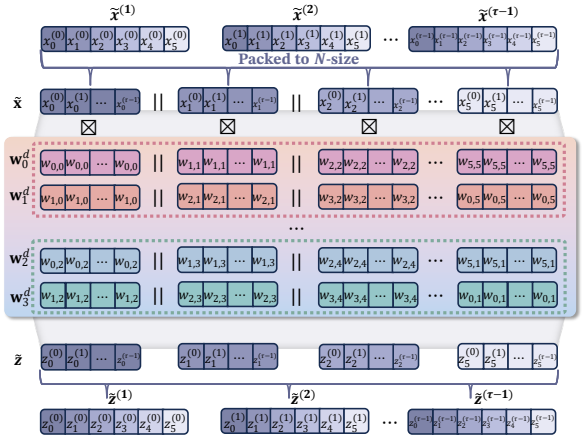


Figure 10: Batched PCMul protocol with group-based packing strategy.

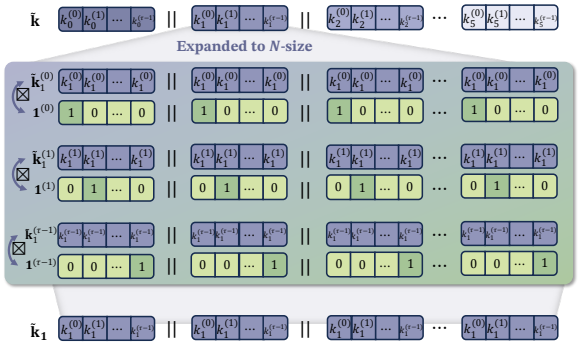


Figure 11: Batched CCMul protocol.

layout of $\tilde{\mathbf{x}}$, each diagonal vector $\boldsymbol{w}_j^d$ in every group is expanded by repeating each element τ times, ensuring that the resulting vector fully occupies the corresponding SIMD slots:

$$\mathbf{w}_j^d \leftarrow \underbrace{w_{j,1}^d, \cdots, w_{j,1}^d}_{\tau} || \cdots || \underbrace{w_{j,m}^d, \cdots, w_{j,m}^d}_{\tau} \tag{19}$$

Within each group, we perform $e-1$ rotations on the packed vector $\tilde{\mathbf{x}}$, *i.e.*, $\mathsf{Rot}(\tilde{\mathbf{x}}, \tau j), j \in [0, e)$, and multiply each rotated vector with its corresponding $\mathbf{w}_j^d$ to obtain intermediate results $\tilde{\mathbf{z}}_j$. Finally, the outputs of all groups are aggregated by applying rotations of stride $\tau \cdot ie$ and summing them:

$$\tilde{\mathbf{z}} = \sum_{i=0}^{g-1} \mathsf{Rot}(\tilde{\mathbf{z}}_i, \tau \cdot ie) \tag{20}$$

yielding the final batched linear-transformation result $\tilde{\mathbf{z}}$.

B.2 Ciphertext-Ciphertext Multiplication

After obtaining the Attention inputs $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v} \leftarrow \boldsymbol{x} \times \boldsymbol{W}_{\{Q,K,V\}}$ via plaintext-ciphertext multiplication, Whisper replaces the standard Self-Attention with Linear-Attention to improve parallel processing capability. Unlike Self-Attention—which computes pairwise token interactions and normalizes them via Softmax—Linear-Attention maintains a continuously updated state vector $\boldsymbol{S}$ to perform weighted aggregation. A key property is that each state update depends only on the current token n, eliminating the need for global normalization over all previous tokens:

$$\boldsymbol{o}_n = \boldsymbol{S}_n \boldsymbol{q}_n, \boldsymbol{S}_n = D\boldsymbol{S}_{n-1} + \boldsymbol{v}_n \boldsymbol{k}_n^\top \tag{21}$$

In the encrypted setting, implementing Linear-Attention requires computing the outer product between $\tilde{\boldsymbol{v}}$ and $\tilde{\boldsymbol{k}}$ under encryption (ciphertext-cpihertext multiplication, CCMul). This operation is efficiently supported by the homomorphic primitive Expn, which expands every element of $\tilde{\boldsymbol{k}}$ into an equal-length vector, enabling direct execution of the required outer-product computation with $\tilde{\boldsymbol{v}}$.

Moreover, a key advantage of this computation pattern is its ability to update the Attention states $\boldsymbol{S}$ for multiple prompts in parallel. When the packed vector $\tilde{\mathbf{k}}$ contains tokens originating from different prompts:

$$\tilde{\mathbf{k}} \leftarrow \mathsf{Enc}(k_0^{(0)}, \cdots, k_0^{(\tau-1)} || \cdots || k_{m-1}^{(0)}, \cdots, k_{m-1}^{(\tau-1)}) \tag{22}$$

we can reorganize the expanded vectors produced by Expn using block-wise one-hot masking, as illustrated in Fig. 11. This reorganization aligns the expanded structure with the multi-prompt packing layout.

Multiplying the reconstructed vectors with the correspondingly packed $\mathbf{v}$ then yields all prompt-specific state updates $\mathbf{S}$ in a single operation. This mechanism enables the system to perform Attention updates for multiple prompts simultaneously, substantially improving throughput in highly concurrent settings.

B.3 Secure Attention

Building on the implementations of the linear protocols PCMul and CCMul, along with their corresponding batched variants, we construct a Secure Attention protocol that supports concurrent processing of multiple prompts. To further improve efficiency, we incorporate an offline–online execution paradigm into the protocol design. The complete protocol is presented in Protocol 1.

Protocol 1: Secure Attention

Input: $\mathcal{C}$ holds input $\mathbf{x} \in \mathbb{C}^{1\times\tau m}$ batched τ prompts; $\mathcal{S}$ holds weight $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{C}^{\tau m\times\tau m}$.

Output: $\mathcal{S}$ holds SIMD packed ciphertext vector $\tilde{\mathbf{z}}$, where $\mathbf{z} \leftarrow \mathsf{Attention}(\mathbf{x})$.

§ **Offline**

for *each matrix* $\mathbf{W} \in \{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V\}$ **do**
  $\mathcal{C}$ samples random vector $\mathbf{a}$ for $\mathbf{x}$ and different weight matrix.
  $\mathcal{C}$ encrypts each random vector after SIMD encoding to obtain $\tilde{\mathbf{a}} \leftarrow \mathsf{Enc}(\mathbf{a})$, and then transmits them to $\mathcal{S}$.
  $\mathcal{S}$ computes $\tilde{\mathbf{b}} \leftarrow \mathsf{PCMul}(\tilde{\mathbf{a}}, \mathbf{W})$ to obtain the ciphertext mask $\tilde{\mathbf{b}}$.
end

§ **Online**

for *each matrix* $\mathbf{W} \in \{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V\}$ **do**
  $\mathcal{C}$ computes $\mathbf{c} \leftarrow \mathbf{x} - \mathbf{a}$ and sends to $\mathcal{S}$.
end
$\mathcal{S}$ computes linear transformation:

$$\tilde{\mathbf{q}} \leftarrow (\mathbf{c} \times \mathbf{W}_Q) \boxplus \tilde{\mathbf{b}};$$
$$\tilde{\mathbf{k}} \leftarrow (\mathbf{c} \times \mathbf{W}_K) \boxplus \tilde{\mathbf{b}};$$
$$\tilde{\mathbf{v}} \leftarrow (\mathbf{c} \times \mathbf{W}_V) \boxplus \tilde{\mathbf{b}};$$

$\mathcal{S}$ updates $\tilde{\mathbf{S}} := D \boxtimes \tilde{\mathbf{S}} \boxplus \mathsf{CCMul}(\tilde{\mathbf{k}}^\top, \tilde{\mathbf{v}})$.
$\mathcal{S}$ executes $\{\tilde{\mathbf{q}}_0, \tilde{\mathbf{q}}_1, \ldots, \tilde{\mathbf{q}}_{k-1}\} \leftarrow \mathsf{Expn}(\tilde{\mathbf{q}})$.
for *each row-wise vector* $\tilde{\mathbf{s}}_j, j \in \{0, k-1\}$ *of* $\tilde{\mathbf{S}}$ **do**
  $\mathcal{S}$ executes $\tilde{\mathbf{z}} \leftarrow \boxplus_{j=0}^{k-1}(\tilde{\mathbf{q}}_j \boxtimes \tilde{\mathbf{s}}_j)$.
end
return $\tilde{\mathbf{z}}$.

C Details of Non-linear Protocol

C.1 Comparison

In fixed-point HE schemes, each ciphertext multiplication increases the underlying plaintext scale by a factor of Δ. To compensate for this scale growth, HE systems employ a modulus chain:

$$q_\ell \leftarrow q_0 \cdot (Q_1 \cdot Q_2 \cdots Q_{\ell-1}) \tag{23}$$

where each rescale step (after a multiplication) removes the highest modulus level, effectively reducing the ciphertext modulus and restoring the scale. Once the chain is fully consumed and only the base modulus q_0 remains, no further ciphertext multiplications are possible. At this point, bootstrapping (Bts) must be invoked to replenish the modulus budget and restore the available multiplicative depth.

In standard Bts of CKKS, the system first performs ModRaise, lifting the ciphertext $\tilde{x} \in \mathcal{R}_q^2$ into a larger modulus $q' \gg q$. This operation embeds $\tilde{x}$ into the extended modulus domain—often expressed as a centered lifting such as $\tilde{x} + qI$, thereby expanding the representable interval from $[-\frac{1}{2}q, \frac{1}{2}q]$ to $[-\frac{1}{2}q', \frac{1}{2}q']$. The Bts procedure then applies a modulo reduction $(\mathrm{mod}\ q)$ to ensure the decrypted value falls back into the correct range. However, modular reduction is inherently periodic and discontinuous, whereas HE natively supports only addition and multiplication—*i.e.*, continuous polynomial operations. Consequently, existing schemes (Cheon et al., 2018; Jutla and Manohar, 2022) exploit the periodic structure of trigonometric functions to rewrite the modulo operation as a composition of smooth periodic functions, which can then be approximated via Taylor series expansions and evaluated homomorphically (see Fig. 12).

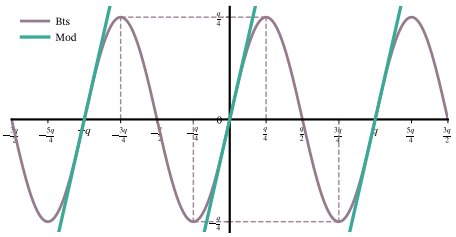


Figure 12: Implementing the Mod function via the periodicity of the Sin function.

In LLMs inference under HE, Bts commonly appears adjacent to non-linear operations such as Cmp for GeLU, which themselves are implemented through polynomial approximations.

$$\mathsf{GeLU}\,(x) = \begin{cases} 0 & \text{if } x < -\phi_0 \\ P_0(x) & \text{if } -\phi_0 < x \le -\phi_1 \\ P_1(x) & \text{if } -\phi_1 < x \le \phi_2 \\ x & \text{if } x > \phi_2 \end{cases} \tag{24}$$

Therefore, if both Bts and these non-linear functions can be expressed within the same trigonometric-based approximation framework, they can be fused into a single homomorphic evaluation. This functional integration reduces computational overhead and significantly improves overall efficiency.

To enable such fusion, we first observe that the decision boundary of Cmp depends solely on the sign of its input. Meanwhile, HE bootstrapping naturally involves evaluating functions in the modular domain. Leveraging this observation, we exploit the periodicity of the sine function to express the sign of x modulo q, thereby embedding Cmp into the same functional space used for Bts, as CBts:

$$\mathsf{Cmp}(x + qI\ (\mathrm{mod}\ q)) = \frac{1 + \mathsf{Sign}(\sin(\frac{2\pi}{q}x))}{2} \tag{25}$$

This formulation leverages the fact that:

- $\sin\left(\frac{2\pi}{q}x\right)$ has the same sign as x within any principal interval of length q.
- enabling Cmp to be computed directly on wrapped ciphertext values during Bts.

Thus, both operations can be executed in a single homomorphic evaluation pipeline, reducing multiplicative depth and eliminating a full additional Bts cycle.

To homomorphically evaluate Eq. 25, we approximate the Sign function using its well-known Fourier sine–series. For any odd, periodic function with period 2π, the Fourier series contains only odd sine terms:

$$\mathsf{Sign}(\sin\theta) = \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\sin((2k-1)\theta)}{2k-1}, \quad \theta = \frac{2\pi}{q}x \tag{26}$$

Substituting Eq. 26 into Eq. 25 yields:

$$\mathsf{Cmp}(x+qI\ (\mathrm{mod}\ q)) = \frac{1}{2}+\frac{2}{\pi}\sum_{k=1}^{\infty}\frac{\sin(\frac{2\pi(2k-1)}{q}x)}{2k-1} \tag{27}$$

To make the computation HE-compatible, the infinite series must be truncated. The degree-N approximation is:

$$\mathsf{Cmp}_N(x+qI\ (\mathrm{mod}\ q)) = \frac{1}{2}+\frac{2}{\pi}\sum_{k=1}^{N}\frac{\sin(\frac{2\pi(2k-1)}{q}x)}{2k-1} \tag{28}$$

where each sine term can be further approximated using its Taylor expansion.

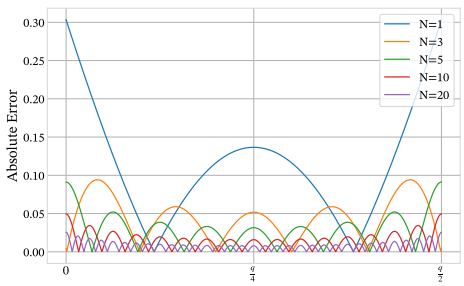


Figure 13: Absolute error of the Cmp function under different approximation degrees.

C.2 Truncation Error Analysis

Theorem 2 (Truncated Cmp Error Bound) *Let*

$$E_N(x) = \mathsf{Cmp}(x) - \mathsf{Cmp}_N(x) \tag{29}$$

denote the truncation error of the Cmp *function approximated by the first N terms of its Fourier sine expansion. Then the absolute error satisfies:*

$$\begin{aligned}|E_N(x)| &\le \frac{4}{\pi}\sum_{k=N+1}^{\infty}\frac{1}{2k-1} \\ &\approx \frac{4}{\pi}\left(\log N + \gamma + O(1/N)\right)\end{aligned} \tag{30}$$

where γ is the Euler–Mascheroni constant. Moreover, over any compact interval $[a,b]$ that excludes the discontinuity points, the convergence is uniform and improves to:

$$|E_N(x)| = O\left(\frac{1}{N}\right), \quad x \in [a,b], b < a+q. \tag{31}$$

Proof 2 *The* Cmp *function is derived from a square-wave–like* Sign *function. Its truncated Fourier sine expansion takes the form*

$$\mathsf{Cmp}_N(x) = \frac{1}{2} + \frac{2}{\pi}\sum_{k=1}^{N}\frac{\sin(\frac{2\pi(2k-1)}{q}x)}{2k-1} \tag{32}$$

By standard Fourier analysis of odd periodic square-wave functions, the truncation error of the series satisfies

$$\begin{aligned}|E_N(x)| &= \left|\frac{2}{\pi}\sum_{k=N+1}^{\infty}\frac{\sin((2k-1)x)}{2k-1}\right| \\ &\le \frac{4}{\pi}\sum_{k=N+1}^{\infty}\frac{1}{2k-1}.\end{aligned} \tag{33}$$

Using the asymptotic expansion of the partial sum of the harmonic series over odd indices yields

$$\sum_{k=N+1}^{\infty}\frac{1}{2k-1} = \log N + \gamma + O(1/N) \tag{34}$$

which establishes the global error bound. On any compact interval excluding discontinuity points, classical Fourier convergence theory ensures uniform convergence of the sine series with rate $\mathcal{O}(1/N)$, proving the local bound.

As illustrated in Fig. 13, within the periodic interval $[0, \frac{1}{2}q]$, the minimum approximation error between our CBts operation and the plaintext Cmp operation remains bounded within $[1.4 \times 10^{-5}, 1.8 \times 10^{-4}]$ across different expansion orders N, and this approximation error does not accumulate across layers. Near the discontinuity ($x = 0 \pmod q$), the series exhibits the classical Gibbs phenomenon; however, these points correspond to a measure-zero region and do not affect correctness in HE inference, where inputs are approximate floating-point ciphertext encodings.

Protocol 2: Secure GeLU

Input: $\mathcal{S}$ holds SIMD packed ciphertext vector input $\tilde{\mathbf{x}}$.

Output: $\mathcal{S}$ holds SIMD packed ciphertext vector $\tilde{\mathbf{z}}$, where $\mathbf{z} \leftarrow \mathsf{GeLU}(\mathbf{x})$.

1 Compare x with the breakpoints:
$\tilde{\mathbf{b}}_0 \leftarrow \mathsf{CBts}(\tilde{\mathbf{x}} \boxplus \phi_0)$ // $b_0 = 1\{x > -\phi_0\}$;
$\tilde{\mathbf{b}}_1 \leftarrow \mathsf{CBts}(\tilde{\mathbf{x}} \boxplus \phi_1)$ // $b_1 = 1\{x > -\phi_1\}$;
$\tilde{\mathbf{b}}_2 \leftarrow \mathsf{CBts}(\tilde{\mathbf{x}} \boxminus \phi_2)$ // $b_2 = 1\{x > \phi_2\}$;

2 Compute segment selection:
$\tilde{\mathbf{s}}_0 \leftarrow 1 \boxminus \tilde{\mathbf{t}}_0$ // $s_0 = 1\{x < -\phi_0\}$;
$\tilde{\mathbf{s}}_1 \leftarrow \tilde{\mathbf{b}}_0 \boxminus \tilde{\mathbf{b}}_1$ // $s_1 = 1\{-\phi_0 < x < -\phi_1\}$;
$\tilde{\mathbf{s}}_2 \leftarrow \tilde{\mathbf{b}}_1 \boxminus \tilde{\mathbf{b}}_2$ // $s_2 = 1\{-\phi_1 < x < \phi_2\}$;
$\tilde{\mathbf{s}}_3 \leftarrow \tilde{\mathbf{b}}_2$ // $s_3 = 1\{x > \phi_2\}$;

3 Compute GeLU:
$\tilde{\mathbf{z}} \leftarrow (\tilde{\mathbf{s}}_0 \boxtimes 0) \boxplus (\tilde{\mathbf{s}}_1 \boxtimes P_0(\tilde{\mathbf{x}})) \boxplus (\tilde{\mathbf{s}}_2 \boxtimes P_1(\tilde{\mathbf{x}})) \boxplus (\tilde{\mathbf{s}}_3 \boxtimes \tilde{\mathbf{x}})$;

4 **return** $\tilde{\mathbf{z}}$.

C.3 Secure GeLU

Following most existing works that implement GeLU via the piecewise polynomial approximation shown in Eq. 24, we design a Secure GeLU protocol using CBts, which integrates both the Bts and Cmp operations. The full protocol is presented in Protocol 2.

D Details of Experiments

D.1 Evaluation Setting

All experiments were conducted on a machine equipped with a 64-core CPU and 128GB RAM. Each experiment was repeated 10 times, and the reported results are averages. The detailed experimental settings are described below.

- For the HE hyperparameters, we set the slot size to 2^{15}, enabling 21 levels of multiplicative depth and providing 128-bit security.
- For performance evaluation, we fine-tune a pre-trained BERT$_{base}$ model and assess its downstream performance on the GLUE benchmark. In the Whisper setting, to simulate a cloud-based teacher model, we assume that the teacher has already learned the target dataset and generates a fine-tuned dataset via PPI to enable PPFT. In contrast, noise-injection–based baselines assume direct access to the fine-tuning samples and perform model fine-tuning on the cloud.
- In the efficiency evaluation, we conduct fine-tuned dataset generation experiments using BERT$_{base}$ and GPT-2$_{medium}$. The inputs consist of 128 prompts, each containing 128 tokens; correspondingly, the outputs are the inference results for 128 prompts, with each output comprising 10 (BERT$_{base}$) or 5 (GPT-2$_{medium}$) tokens.

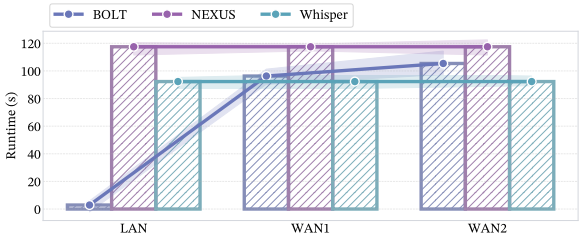


Figure 14: Efficiency Comparison of GeLU on $\mathbb{C}^{2^{20}}$.

D.2 Additional Efficiency Evaluation

Table 3: Efficiency comparison of PCMul on $\mathbb{C}^{k \times m} \times \mathbb{C}^{m \times m}$ ($k = 256, m = 2048$, and Runtime in s).

Framework	#KS	#Mul	Runtime	Factor
BOLT	6656	256	11.14	1.00 ×
NEXUS	8208	524288	20.69	▼1.85 ×
Whisper	3040	256	4.56	▲2.44 ×

Linear. We further evaluate the efficiency of the linear protocol PCMul under large-scale matrix computation scenarios in Table 3. For the NEXUS scheme, additional $\mathcal{O}(\frac{km}{N})$ Expand operations with complexity $\mathcal{O}(\log_2 N)$ are required to unpack the packed ciphertexts, which further introduces a multiplication overhead of $\mathcal{O}(km)$. Although BOLT reduces the theoretical rotation complexity by adopting the BSGS technique, its overall computational cost remains high in large-scale matrix operations, leading to a substantial increase in the number of key-switching and multiplication calls during execution. In contrast, our proposed group-based packing strategy reduces the rotation complexity to $\mathcal{O}(k(\frac{m}{g} + g - 2)/2)$ and, when combined with

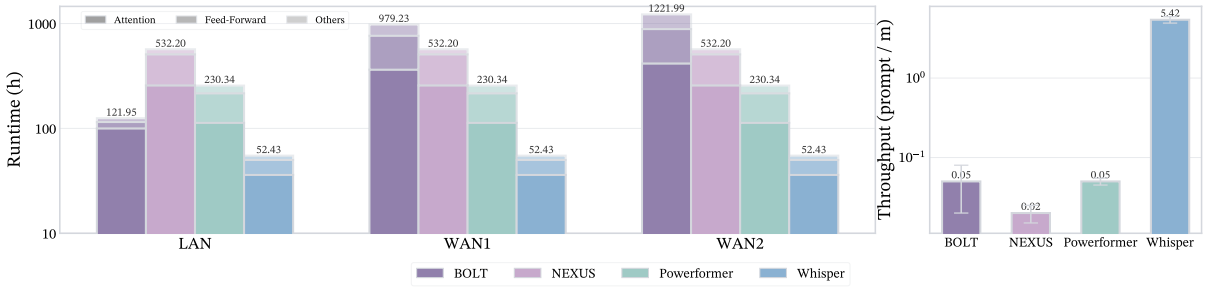


Figure 15: Runtime and throughput for generating a 128-sample fine-tuned dataset under different PPI schemes on GPT-2 model.

hoisting, effectively decreases the number of key-switch operations in practice. As a result, our approach maintains efficiency gains comparable to those observed in small-scale settings even under large-scale matrix computation workloads.

Non-linear. We further evaluate the computational efficiency of the non-linear GeLU protocol under large input lengths in Fig. 14. The BOLT scheme, which implements non-linear operations based on MPC, involves multiple rounds of interaction and is therefore highly sensitive to network communication conditions. In contrast, in the HE-based NEXUS scheme, the Bts and Cmp operations are executed independently, introducing additional computational overhead during polynomial evaluation. By comparison, Whisper adopts a Fourier sine-series approximation and exploits the periodicity of trigonometric functions to fuse the Bts operation with the non-linear approximation Cmp, resulting in the proposed CBts protocol. This design eliminates the need to evaluate two separate polynomials. As a result, Whisper achieves an approximately 27% improvement in runtime efficiency, even in large-input regimes.

Dataset Generation. We further evaluate the end-to-end efficiency of fine-tuned dataset generation on the GPT-2_{medium} model in Fig. 15. Under the same setting where SIMD is used for ciphertext packing, Whisper benefits from the inherent parallelism of the Linear-Attention architecture, enabling the simultaneous processing of multiple prompt requests. In contrast, approaches based on Self-Attention can only execute prompt inference sequentially. In large-scale fine-tuned dataset generation tasks, Whisper integrates protocol optimizations across both linear and nonlinear layers, achieving an 108.40-271.01$\times$ improvement in throughput and reducing runtime by 4.39-23.30$\times$ compared to the SOTA PPI scheme. These results demonstrate that Whisper can generate fine-tuned datasets more efficiently, thereby effectively supporting the practical deployment of PPFT.