SecP-Tuning: Efficient Privacy-Preserving Prompt Tuning for Large Language Models via MPC

Anonymous authors
Paper under double-blind review

Abstract

Large Language Models (LLMs) have revolutionized numerous fields, yet their adaptation to specialized tasks in privacy-sensitive domains such as healthcare and finance remains constrained due to the scarcity of accessible training data caused by stringent privacy requirements. Secure Multi-party Computation (MPC)-based privacy-preserving machine learning provides theoretical guarantees for the privacy of model parameters and data. However, its application to LLMs has been predominantly limited to inference, as fine-tuning introduces significant efficiency challenges, particularly in backward propagation, optimizer, and self-attention operations. To address these challenges, we propose ***SecP-Tuning***, *the first MPC-based framework designed for efficient, privacy-preserving prompt tuning of LLMs*. SecP-Tuning innovatively integrates Forward-only Tuning (FoT) through the "data owner-server interaction" paradigm, effectively removing the need for privacy-preserving computations in backward propagation and optimization processes. Furthermore, it devises an efficient privacy-preserving Random Feature Attention (RFA), effectively mitigating the computational complexity of softmax-based self-attention and circumventing MPC-incompatible nonlinear operations. Experimental results demonstrate that, compared to full-Parameter Supervised Fine-Tuning (SFT) and gradient-based prompt tuning, SecP-Tuning achieves approximately **12**$\times$ and **16**$\times$ end-to-end acceleration, as well as **18**$\times$ and **20**$\times$ reductions in communication overhead, respectively. Moreover, it delivers performance comparable to gradient-based methods across multiple few-shot tasks. Additionally, the "black-box/API-style" privacy-preserving tuning paradigm of SecP-Tuning effectively avoids memory leakage risks caused by gradient/parameter transmission, thereby *striking an optimal balance between efficiency, accuracy, deployability, and privacy.* The code will be released.

1 Introction

Large Language Models (LLMs) (Vaswani et al., 2017; Liu et al., 2019; Hurst et al., 2024; Dubey et al., 2024; Guo et al., 2025) have achieved groundbreaking advancements in diverse domains, including natural language understanding, generation, reasoning, and cross-modal applications. However, adapting universally pre-trained LLMs to high-sensitivity fields such as healthcare, finance, government compliance, and industrial manufacturing remains a significant challenge. This difficulty arises from the fact that such sensitive data is closely tied to the interests of data owners and is subject to regulations (e.g., GDPR, HIPAA) and corporate compliance requirements, making direct access impractical. Additionally, model parameters may encapsulate statistical information from the source domain, posing potential privacy risks. Therefore, the key scientific and engineering challenge in achieving the implementation of "trustworthy intelligence" lies in efficiently adapting LLMs to specific domains using effective fine-tuning methods, such as Full-Parameter Supervised Fine-Tuning (SFT) (Wei et al., 2021; Devlin et al., 2019), Low-Rank Adaptation (LoRA) (Hu et al., 2022; Dettmers et al., 2023), and Prompt Tuning (Lester et al., 2021; Liu et al., 2022), while ensuring that neither the *fine-tuning data* nor the resulting *model parameters* are exposed.

Privacy-Preserving Machine Learning (PPML) based on Secure Multi-Party Computation (MPC) (Yao, 1986; Goldreich et al., 1987) offers a promising solution. In this paradigm, model

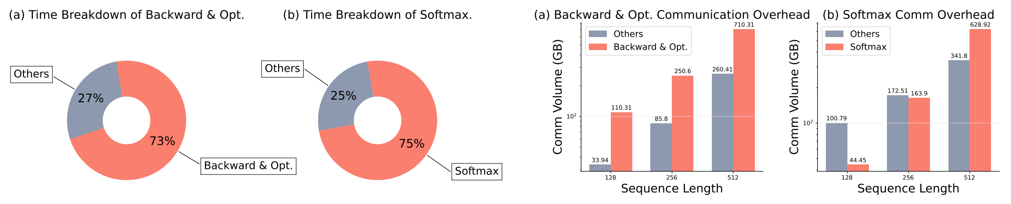

Figure 1: The time breakdown for SFT of RoBERTa$_{\text{LARGE}}$ (24 layers, 1024 dimensions) using MPC is analyzed with a sequence length of 512, along with a comparison of communication volumes across different sequence lengths.

parameters and sensitive data are first secret-shared among participating parties. These parties then execute MPC protocols through multiple rounds of communication to complete privacy-preserving computations for forward propagation, backward propagation, and optimization. All computations are performed on secret-shared inputs and intermediate results, ensuring that parties only learn the protocol's explicitly permitted outputs without accessing private data or model parameters. Due to its compelling privacy guarantees, MPC-based PPML has been successfully applied to the training of linear models (Mohassel and Zhang, 2017), convolutional neural networks Wagh et al. (2019; 2021), and the inference of Transformer-based LLMs (Hao et al., 2022; Luo et al., 2024; Pang et al., 2023; Lu et al., 2023).

However, implementing Privacy-Preserving Fine-Tuning (PFT) of LLMs directly using MPC incurs prohibitive overhead. For instance, performing SFT on RoBERTa$_{\text{LARGE}}$ (Liu et al., 2019), consisting of 24 layers and 1024 dimensions, with a sequence length of 512 requires approximately 10 minutes per iteration and incurs a communication overhead of 970GB over a Local-Area Network (LAN) with 3Gbps bandwidth and 0.8ms latency. As illustrated in Figure 1, two primary factors contribute to this overhead: a) *Backward propagation and optimization*, which account for 73% of the total runtime, far exceeding the cost of forward propagation. This is due to the presence of numerous MPC-unfriendly nonlinear operations in backward propagation and optimization, such as Softmax, GELU, and LayerNorm, which must undergo privacy-preserving reverse computation. These operations cannot be directly executed in MPC environments and must be decomposed into approximations using addition, multiplication, and comparison, leading to a dramatic increase in communication rounds and volume. b) *Softmax in the self-attentio*n, which contributes 75% of the total runtime. This is because Softmax involves a large number of MPC-unfriendly nonlinear operations, including exponentiation, division, and maximum computation. Furthermore, its computational complexity scales quadratically with the input sequence length, causing communication overhead to grow rapidly as sequence length increases. *Gradient-based efficient parameter fine-tuning methods*, such as LoRA and gradient-based prompt tuning, effectively reduce the number of parameters requiring updates and enhance the efficiency of privacy-preserving optimization. However, they *fail to resolve the fundamental communication overhead caused by backward propagation and Softmax operations in MPC settings.*

In this paper, we take the first step toward addressing the research question: ***How to perform privacy-preserving domain adaptation of LLMs in MPC environments efficiently and with high performance?*** Specifically, we propose SecP-Tuning, the first MPC-based privacy-preserving framework for prompt tuning in LLMs. SecP-Tuning leverages *Forward-only Tuning (FoT)* (Sun et al., 2022b;a) to update prompt parameters, fundamentally eliminating the high communication overhead caused by backward propagation in gradient-based fine-tuning methods, thereby significantly accelerating the privacy-preserving adaptation process. To address the MPC-unfriendly loss value and Gradient-Free Optimizer (GFO) (Rios and Sahinidis, 2013) computations in FoT, we introduce an innovative "*Server-Client*" architecture. In this architecture, MPC-unfriendly computations for loss values and GFO are offloaded to the data owner's local environment for efficient and precise plaintext computation. This approach not only significantly improves speed but also prevents the server from accessing updated prompt parameters, thereby mitigating the privacy risks of fine-tuning data leakage caused by model memorization. Complementing this, we propose *privacy-preserving Random Feature Attention (RFA)*, which avoids extensive nonlinear operations in softmax while reducing the complexity of self-attention from quadratic to linear.

The experimental results systematically validate the comprehensive advantages of SecP-Tuning across multiple dimensions, including *efficiency, performance, deployability, and privacy*. Compared to SFT and gradient-based prompt tuning, SecP-Tuning achieves approximately 12$\times$ and 16$\times$ end-to-end acceleration, respectively, while reducing communication volume by about 18$\times$ and 20$\times$. Notably, these acceleration advantages are further amplified in bandwidth-constrained Wide-Area Network (WAN) scenarios. In terms of performance, SecP-Tuning demonstrates superior results on multiple few-shot fine-tuning tasks (16 samples per class), with an average score of 82.45, comparable to SFT's 85.41 and gradient-based cue-based tuning's 83.84. Deployability comparison further highlights that SecP-Tuning supports "black-box/API-style" secure tuning, effectively preventing the potential privacy risks of memory leakage caused by gradient/parameter transmission back to the server.

2 Related Work

Cryptographic techniques such as MPC and Homomorphic Encryption (HE) (Gentry, 2009; Cheon et al., 2017) have been widely applied in privacy-preserving machine learning, including early works on linear networks (Mohassel and Zhang, 2017) and training and inference for convolutional neural networks (Wagh et al., 2019; 2021; Liu et al., 2017; Riazi et al., 2018; Juvekar et al., 2018). With the rise of Transformer-based LLMs, researchers have increasingly focused on privacy-preserving inference for LLMs (Hao et al., 2022; Li et al., 2023; Zeng et al., 2022; Luo et al., 2024; Pang et al., 2023; Yan et al., 2025), aiming to protect both model parameters and inference data. However, compared to inference, fine-tuning LLMs involves complex backward propagation and optimizer computations, which remain underexplored.

Currently, only a few studies perform privacy-preserving domain adaptation of LLMs based on HE. Specifically, the first HE-based PFT framework, BlindTuner (Panzade et al., 2025), enhances practicality through pre-trained feature extraction while maintaining accuracy. Subsequently, Med-BlindTuner (Panzade et al., 2024) extended this approach to biomedical imaging and validated its effectiveness. To further reduce computational overhead, later works introduced parameter-efficient methods like LoRA: PrivTuner (Li et al., 2024b), which integrates LoRA with FHE to reduce computation overhead. Rho et al. (2025) replaced self-attention with Gaussian Kernel Attention to mitigate the costs of nonlinear operations. In addition, FedShield-LLM (Mia and Amini, 2025) reduced computational overhead by combining unstructured pruning techniques.

Unlike HE, which relies on intensive unilateral encryption computations and requires costly approximations and re-encryption for nonlinear operations such as Softmax and GELU, making it difficult to balance efficiency and accuracy, MPC enables complex nonlinear operations through multi-round communication among participants. This makes MPC more suitable for PFT. However, to the best of our knowledge, no prior work has explored MPC-based PFT of LLMs.

In addition to cryptographic techniques, Differential Privacy (DP) Dwork and Roth (2014) has also been applied to privacy-preserving fine-tuning. The primary goal of DP-based privacy-preserving fine-tuning algorithms (Wang et al., 2024; Li et al., 2024a; Charles et al., 2024) is to ensure individual-level privacy. This is achieved by introducing mechanisms such as adding random noise and clipping during the fine-tuning process, which formally limit the influence of any single training sample on the final model. The privacy guarantee is quantified by the (ϵ, δ) privacy budget. In contrast, MPC-based privacy-preserving fine-tuning frameworks provide theoretical privacy guarantees for privacy parameters and fine-tuning data under a specified threat model, which is fundamentally different from DP-based privacy-preserving frameworks.

3 Preliminaries

3.1 Softmax-based Self-Attention & Random Feature Attention

Softmax-based Self-Attention. The core component of each Transformer layer is the self-attention mechanism. We omit a detailed discussion of the feed-forward network and other auxiliary components, as they remain unchanged in our work. Let n and d denote the sequence length and embedding dimension, respectively. The self-attention mechanism is computed as follows:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V} \in \mathbb{R}^{n\times d}\,. \tag{1}$$

Here, the rows of $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ correspond to the query, key, and value vectors. The softmax function (Bridle, 1989) is applied row-wise, converting the similarity scores between each query and all key vectors into a probability distribution that weights the contribution of each value vector.

Random Feature Attention. To speed up the softmax operations in attention, Peng et al. (2021) has employed random feature (Rahimi and Recht, 2007) methods to approximate the dot-then-exponentiate operation using kernel tricks. The main idea is to approximate the Gaussian kernel function via its Monte Carlo estimation:

$$\exp\big(-\|\mathbf{x}-\mathbf{x}')\|^2/\sigma^2\big) \approx \sum\nolimits_{i=1}^{M} \varphi(\mathbf{x},\omega_i)\varphi(\mathbf{x}',\omega_i)\,, \tag{2}$$

where $\varphi(\mathbf{x},\omega_i) = \sqrt{2/M}\cos(\omega_i^\top \mathbf{x} + b_i)$, with $\omega_i \sim \mathcal{N}(0,\sigma^2 I)$ and $b_i \sim U(0,2\pi)$.

Let $\phi(\mathbf{x}) = \exp(\|\mathbf{x}\|^2/(2\sigma^2))\big[\varphi(\mathbf{x},\omega_1),...,\varphi(\mathbf{x},\omega_M)\big]^\top$, the dot-then-exponentiate function can be approximated as:

$$\exp\big(\mathbf{x}^\top\mathbf{y}/\sigma^2\big) = \exp\big(\frac{1}{2\sigma^2}\|\mathbf{x}\|^2 + \frac{1}{2\sigma^2}\|\mathbf{y}\|^2\big)\exp\big(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{y}\|^2\big) \approx \phi(\mathbf{x})^\top\phi(\mathbf{y}). \tag{3}$$

Substituting this approximation into the softmax attention, we obtain the RFA:

$$\begin{aligned}
\text{Softmax}(\mathbf{q}_t, \{\mathbf{k}_i\}_{i=1}^n, \{\mathbf{v}_i\}_{i=1}^n) &= \sum_i \frac{\exp(\mathbf{q}_t^\top \mathbf{k}_i/\sigma^2)\mathbf{v}_i^\top}{\sum_j \exp(\mathbf{q}_t^\top \mathbf{k}_j/\sigma^2)} \\
&\approx \sum_i \frac{\phi(\mathbf{q}_t)^\top \phi(\mathbf{k}_i)\mathbf{v}_i^\top}{\sum_j \phi(\mathbf{q}_t)^\top \phi(\mathbf{k}_j)} \\
&= \frac{\phi(\mathbf{q}_t)^\top \sum_i \phi(\mathbf{k}_i)\otimes \mathbf{v}_i}{\phi(\mathbf{q}_t)^\top \sum_j \phi(\mathbf{k}_j)} := \text{RFA}(\mathbf{q}_t, \{\mathbf{k}_i\}_{i=1}^n, \{\mathbf{v}_i\}_{i=1}^n)\,,
\end{aligned} \tag{4}$$

where $Q = \{\mathbf{q}_i\}_{i=1}^n, K = \{\mathbf{k}_i\}_{i=1}^n, V = \{\mathbf{v}_i\}_{i=1}^n$, and $\otimes$ denotes the outer product between vectors. Leveraging this linearized formulation, RFA achieves linear time and memory complexity with respect to the sequence length.

3.2 Gradient-Free Optimization

Gradient-Free Optimization (GFO) (Rios and Sahinidis, 2013) optimizes an objective using only function (fitness) evaluations, without gradients; hence, it is also called black-box or zeroth-order optimization. These methods follow a sample–evaluate–update loop and are well-suited to settings where derivatives are unavailable or too expensive. Black-Box Tuning (Sun et al., 2022b) applies GFO to prompt tuning for large language models (LLMs), learning a continuous prompt vector $p \in \mathbb{R}^D$ that minimizes $p^* = \arg\min_{p\in\mathcal{P}} \mathcal{L}\big(f(p;X),Y\big)$, where f is the LLM inference function, $\mathcal{L}$ the loss, and $\mathcal{P}$ the prompt space. Because GFO convergence typically degrades in high dimensions, BBT exploits the low intrinsic dimensionality of LLM prompts by optimizing a latent variable $z \in \mathbb{R}^d$ with $d \ll D$ and mapping it via a random projection $A \in \mathbb{R}^{D\times d}$:

$$z^* = \arg\min_{z\in\mathcal{Z}} \mathcal{L}\big(f(Az;X),Y\big). \tag{5}$$

CMA-ES (Hansen, 2016) is used in this paper as the gradient-free optimizer.

3.3 2-out-of-2 Arithmetic Secret Sharing

For an integer ring $\mathbb{Z}_n = \{0,1,\ldots,n-1\}$, a 2-out-of-2 arithmetic secret sharing scheme involves the following two algorithms:

- The sharing algorithm $Shr(x) \to ([x]_0,[x]_1)$ is used to generate the shares of x. Specifically, a value r is chosen *uniformly at random* from $\mathbb{Z}_n$, such that $[x]_0 = r$, and $[x]_1 = x - r \pmod{n}$ is computed.

- The reconstruction algorithm $Rec([x]_0, [x]_1) \to x$ is used to reconstruct x, i.e., $x = [x]_0 + [x]_1 \pmod{n}$.

The randomness and uniformity of the share ensure that any individual share reveals no information about the secret. We denote the arithmetic secret sharing of x as $[x] = ([x]_0, [x]_1)$.

In the field of secure MPC, numerous secure protocols have been developed for operating over secret shares $[x]$, including secure addition, multiplication, comparison, and various nonlinear activation functions. These cryptographic primitives are summarized in Section 6.3. In this work, we treat these primitives as black-box components and utilize them without requiring additional assumptions or modifications.

4 SecP-Tuning

4.1 MPC-based Privacy-preserving Fine-tuning

The objective of privacy-preserving fine-tuning based on MPC is to fine-tuning a model while safeguarding the privacy of both the developer's proprietary model parameters and the data owner's privacy data, ultimately producing fine-tuned parameters. This process involves two principal parties: the *model developer* and the *data owner*. The model developer possesses a proprietary model F_Θ, where Θ represents private parameters, while the data owner holds confidential fine-tuning data X. In this framework, both parties provide the shares of F_Θ and X, namely $([\Theta]_0, [\Theta]_1)$ and $([X]_0, [X]_1)$, as inputs. These shares are processed using various two-party MPC protocols, such as privacy-preserving addition, multiplication, and GeLU activation functions, to perform privacy-preserving inference and generate the shares of the fine-tuned parameters. Under well-defined threat models such as semi-honest and malicious models, the theoretical security is guaranteed by MPC protocols and ensures the following: 1) Confidentiality of the model developer's parameters; 2) Confidentiality of the data owner's fine-tuning data; and 3) Confidentiality of the fine-tuned parameters.

For efficiency considerations, we adopt the *semi-honest* threat model.[1] In the semi-honest model, participants execute each step of the protocol correctly and obtain accurate results but may attempt to infer unauthorized information during execution. The semi-honest threat model is widely used in Privacy-Preserving Machine Learning (PPML), including early works on privacy-preserving convolutional neural network training and more recent efforts in privacy-preserving inference for LLMs.

Although MPC-based privacy-preserving fine-tuning provides theoretical assurances for the privacy of model parameters and fine-tuning data while *achieving performance comparable to plaintext computation*, directly employing MPC for fine-tuning faces significant efficiency challenges. These challenges primarily stem from the computational costs associated with executing privacy-preserving backpropagation, optimizers, and self-attention mechanisms using MPC. To address these issues, we propose *SecP-Tuning*, which leverages the intrinsic properties of MPC protocols and *incorporates custom-designed, modular components to significantly enhance the efficiency of privacy-preserving fine-tuning.*

4.2 Privacy-preserving Forward-Only Tuning

During the backpropagation phase, numerous nonlinear operators, such as Softmax, GELU, and LayerNorm, must undergo privacy-preserving reverse computation. In the MPC environment, these operations cannot be executed directly and must instead be decomposed into fundamental operations like addition, multiplication, and comparison for approximate computation. This decomposition significantly amplifies both the number of communication rounds and the overall communication volume. Furthermore, the deeply stacked architecture of Transformers exacerbates these costs. Additionally, frequent tensor transpositions, dimension rearrangements, and mask handling during gradient computation, which are mere memory operations in plaintext, require explicit arithmetic-to-Boolean domain conversions and additional synchronization in MPC environments, further increasing communication overhead.

[1]While the malicious threat model better aligns with real-world scenarios, its computational overhead is significantly higher than that of the semi-honest model. Typically, additional cryptographic techniques such as zero-knowledge proofs are required to enhance the semi-honest model.

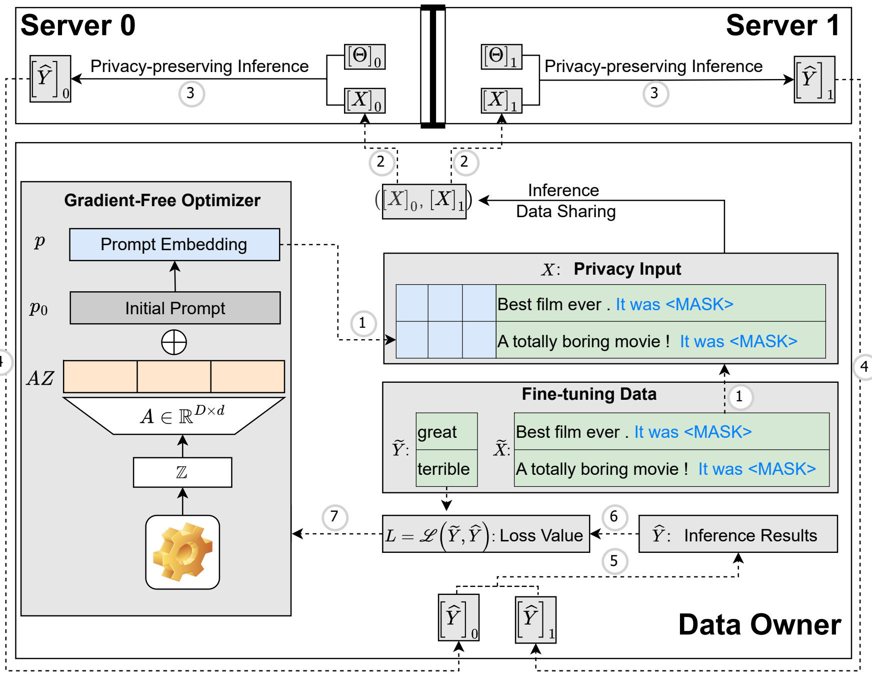
Server 0
Server 1
Privacy-preserving Inference
Privacy-preserving Inference
$[\hat{Y}]_0$
$[\hat{Y}]_1$
$[\Theta]_0$
$[X]_0$
$[\Theta]_1$
$[X]_1$
$([X]_0, [X]_1)$
Inference Data Sharing
Gradient-Free Optimizer
p Prompt Embedding
p_0 Initial Prompt
AZ
$A \in \mathbb{R}^{D \times d}$
$\mathbb{Z}$
X: Privacy Input
Best film ever . It was <MASK>
A totally boring movie ! It was <MASK>
Fine-tuning Data
$\tilde{Y}$: great terrible
$\tilde{X}$:
$L = \mathscr{L}(\tilde{Y}, \hat{Y})$: Loss Value
$\hat{Y}$: Inference Results
Data Owner

Figure 2: Workflow of SecP-Tuning. SecP-Tuning leverages secure MPC to protect both training data and model parameters during fine-tuning. It addresses two key bottlenecks in PFT. First, it eliminates the computational overhead of backward and optimizer by adopting a FoT paradigm. Second, it improves the efficiency of privacy-preserving self attention by employing RFA.

During the optimization phase, widely used optimizers like Adam (Kingma and Ba, 2015) require numerous element-wise operations, including multiplication, division, square root computation, and bias correction, to perform parameter updates. Among these, division and square root computations are particularly costly in MPC environments. Moreover, weight decay, learning rate scheduling (e.g., cosine, multi-stage, or adaptive scheduling), and gradient scaling (used in mixed-precision simulations) introduce additional nonlinear operations and conditional branching. These complexities compel frequent domain conversions between arithmetic and boolean fields in MPC environments, resulting in substantial communication overhead.

Forward-only Tuning (FoT) updates parameters via GFO, fundamentally circumventing the high communication overhead caused by privacy-preserving backpropagation in gradient-based fine-tuning methods. This presents a promising avenue for enhancing the efficiency of privacy-preserving fine-tuning. However, unlike gradient-based optimizers such as Adam, *GFO methods, such as CMA-ES, often involve complex operations that are unable to support in MPC-based PPML frameworks*, such as CrypTen. These operations include *ranked index order*, *outer product of vectors*, and *matrix eigendecomposition*. This hinder the development of an MPC-based privacy-preserving FoT.

To address this issue, SecP-Tuning integrates the features of MPC and FoT to design a "*Server-Client*" architecture that ensures privacy while offloading GFO and loss computation to the client for plaintext processing. This approach not only significantly enhances efficiency but also prevents the server from accessing the updated prompt embeddings, thereby mitigating the risk of fine-tuning data privacy leakage caused by model memorization.

As shown in Fig. 2, SecP-Tuning consists of the following seven steps: 1) The data owner locally initializes the prompt embedding p and concatenates it with the private fine-tuning token embedding to obtain the private input embedding X; 2) The data owner locally generates secret shares of X, denoted as $([X]_0, [X]_1)$, and distributes them to the corresponding servers; 3) Two non-colluding servers take $([X]_0, [X]_1)$ and the secret shares of the private model parameters, $([\Theta]_0, [\Theta]_1)$, as inputs. They interactively execute privacy-preserving inference using MPC protocols, producing secret shares of the inference result $([Y]_0, [Y]_1)$; 4) The servers send $([Y]_0, [Y]_1)$ back to the data owner; 5) The data owner reconstructs the inference result Y using $([Y]_0, [Y]_1)$; 6) The data owner takes the inference result Y and the fine-tuning data labels $\tilde{Y}$ as inputs and calculates the loss value L locally in plaintext; 7) The data owner inputs the loss value L into the GFO to update the prompt embedding. By iterating this process multiple times, the data owner ultimately obtains the fine-tuned prompt embedding for privacy-preserving downstream task inference.

SecP-Tuning leverages the FoT framework from (Sun et al., 2022b) to implement privacy-preserving fine-tuning. To guarantee fairness and reproducibility of results, it adopts the same GFO, CMA-ES. However, readers are free to select other gradient-free optimizers, such as random search, Natural Evolution Strategies, or Bayesian optimization, based on the specific requirements of their scenarios, thereby further enhancing the flexibility and adaptability of SecP-Tuning.

4.3 PRIVACY-PRESERVING RANDOM FEATURE ATTENTION

Although privacy-preserving FoT based on "*Server-Client*" architecture addresses the overhead of privacy-preserving computation in backpropagation and optimizers, SecP-Tuning still faces severe efficiency challenges stemming from the privacy-preserving implementation of softmax-based self-attention mechanisms. Specifically, for a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, Softmax in Transformer converts it to an n-dimensional probability distribution with

$$\text{Softmax}(\mathbf{x})[i] = \frac{e^{x_i - \tau}}{\sum_{h=1}^{n} e^{x_h - \tau}}, \tag{6}$$

where $\tau = \max\left(\{x_h\}_{h=1}^{n}\right)$ is used to ensure stable numerical computations.

There are the following challenges in performing privacy-preserving computation on softmax-based self-attention:

- **Quadratic complexity with respect to sequence length.** Given $(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{n \times d}$, where n denotes the sequence length and d the embedding dimension, the complexity of Softmax-based attention scales as $O(n^2 d)$. This quadratic dependence becomes prohibitively expensive for long input sequences.
- **Numerous nonlinear operations incompatible with MPC.** As shown in Eq. (6), computing the Softmax function involves three nonlinear operations—exponentiation, division, and maximization—all of which are costly to implement under MPC. These operations significantly inflate the overhead of privacy-preserving attention computation (see Section 6.3.2 for details).

To tackle these challenges, *SecP-Tuning employs Random Feature Attention (RFA) to enhance the efficiency of privacy-preserving Softmax-based self-attention mechanisms.* Specifically, compared to existing softmax approximation methods (Kitaev et al., 2020; Wang et al., 2020; Roy et al., 2021), RFA offers the following advantages:

- **Theoretical Guarantee on Approximation Error.** The approximation error is formally bounded, ensuring reliable accuracy.
- **Reduction in Computational Complexity of Softmax.** RFA reduces the complexity of softmax attention from $O(n^2 d)$ to $O(ndr)$, where r represents the number of random features used.
- **Avoidance of Exponentiation and Maximum Operations in Softmax.** By bypassing these costly nonlinear operations, RFA significantly improves efficiency in privacy-preserving settings.

According to Eq. (4), the computation of RFA involves multiplication, division, and cosine function operations. This implies that although RFA bypasses the exponential and maximum operations in softmax-based attention, it introduces *cosine* operations that are not friendly to MPC.

To address this challenge, SecP-Tuning design an efficient MPC-based privacy-preserving cosine function protocol (Π_{cosine}) by leveraging the periodicity of trigonometric functions and the sum-to-product formulas. By executing Π_{cosine}, MPC participants can compute the shares of the result $y = \cos(x)$ while preserving the privacy of the input data x. Specifically, Π_{cosine} consists of two phases: an *offline phase* and an *online phase*. In the offline phase, the computation servers $S_j, j = 0, 1$, pre-generate random numbers $t \in Z_L$ and shares of $\sin(t)$, $\cos(t)$, and t, denoted as $([t]_j, [\sin(t)]_j, [\cos(t)]_j)$. During the online phase, server S_j initially computes $[\delta]_j = [x]_j + [t]_j$. Subsequently, $\delta = (x + t) \mod \tau$, where τ represents the periodicity of the trigonometric function, is reconstructed through a single round of bidirectional communication. Finally, each server S_j computes the shares of $\cos(x)$ using the trigonometric addition identity formulas, $\cos(x) = \sin(\delta)\sin(t) + \cos(\delta)\cos(t)$.

By executing Π_{cosine}, the privacy-preserving computation of the cosine function can be accomplished with only a single round of communication, transmitting 2ℓ-bit elements. Building upon this result, we further develop an efficient MPC-based privacy-preserving RFA protocol, which reduces the computational complexity of the Softmax-based attention mechanism while circumventing the need for expensive exponentiation and maximum operations. Detailed algorithmic descriptions are provided in Section 6.4.

5 Experiments

5.1 Setup

MPC-Backend & Testbeds. Our implementation is based on the PPML framework CrypTen [2], while the execution of FoT and RFA relies on the open-source libraries provided in (Sun et al., 2022b) and (Peng et al., 2021). We conduct our experimental evaluations on three servers, each equipped with an A100 GPU. To enable a comprehensive efficiency comparison, we utilize *Linux Traffic Control (TC)* to simulate various network conditions. Specifically: In the LAN scenario, we set the bandwidth to 3 Gbps with a round-trip latency of 0.8 ms. For the WAN setting, we consider two different configurations: {100 Mbps, 80 ms} and {200 Mbps, 40 ms}.

Model and Dataset. We select RoBERTa$_{\text{LARGE}}$ as the backbone model to validate the effectiveness of SecP-Tuning across five representative datasets: SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), RTE Wang et al. (2018), Yelp Polarity (Zhang et al., 2015), and AG's News (Zhang et al., 2015). To ensure the reproducibility of experimental results, we adopt the same hyperparameter settings as (Sun et al., 2022b) for FoT execution. For RFA, we follow the initialization settings from (Peng et al., 2021) and set the number of random features r to 128. Detailed configurations are provided in Section 6.6.2 of the appendix.

Baselines. To demonstrate the effectiveness of SecP-Tuning, we established the following baselines: 1) SFT: Supervised fine-tuning of all model parameters of pre-trained model. 2) Prompt Tuning: Training only the prompt embeddings added to the input text while keeping the pre-trained model parameters frozen. For a fair comparison, we used the same prompt length, manual templates, label words, and pre-trained prompt embeddings as SecP-Tuning during initialization. We explored a wide range of learning rates and implemented an early stopping mechanism to prevent overfitting of gradient-based methods in few-shot scenarios. Specifically, for SFT, the learning rates were set to [1e-6, 3e-6, 5e-6, 1e-5, 3e-5, 5e-5, 1e-4], with a maximum of 200 epochs and an early stopping patience of 30 steps. For gradient-based Prompt Tuning, the learning rates were set to [1e-5, 3e-5, 5e-5, 1e-4, 3e-4, 5e-5, 1e-3], with a maximum of 1000 epochs and an early stopping patience of 50 steps. See Section 6.6.1 of the appendix for specific hyperparameter and corresponding configurations.

Table 1: Efficiency Comparison of RoBERTa$_{\text{LARGE}}$ in LAN Setting (3Gbps, 0.8ms). The input sequence length is set to 512, and the number of prompt tokens is set to 50. The results are the average of ten runs.

Methods	Forward		Backward		Optimizer		Total	
	Times(s)	Comm(GB)	Times(s)	Comm(GB)	Times(s)	Comm(GB)	Times(s)	Comm(GB)
SFT	216.184	260.411	554.512	691.150	20.902	19.159	651.598	970.720
Prompt Tuning	273.313	306.711	605.212	804.900	3.550	4.594	882.075	1116.205
SecP-Tuning (FoT)	173.999	205.358	0.000	0.000	0.138	0.000	174.138	205.359
SecP-Tuning (FoT+RFA)	**54.17**	**56.545**	**0.000**	**0.000**	**0.103**	**0.000**	**55.172**	**56.545**

[2]https://github.com/facebookresearch/CrypTen

5.2 EFFICIENCY COMPARISON

We perform an end-to-end execution of SecP-Tuning on CrypTen and compare it against baseline methods. To ensure fairness, all executions use CrypTen's privacy-preserving operations and default settings[3]. Table 1 shows the time and communication overhead of different methods in a LAN environment, with additional results in a WAN environment provided in Section 6.2. Compared to SFT and gradient-based prompt tuning, SecP-Tuning delivers substantial advancements in both fine-tuning speed and communication efficiency. Specifically, in a LAN environment, SecP-Tuning achieves a 12 times speedup over SFT and a 16 times speedup over gradient-based prompt tuning. Additionally, it reduces communication overhead by 18 times and 20 times, respectively. This is primarily attributed to SecP-Tuning 's innovative integration of FoT through the "data owner-server interaction" paradigm, which eliminates privacy-preserving computations for backward propagation and optimization. Additionally, the privacy-preserving protocol Π_{RFA} proposed in this paper significantly enhances the efficiency of self-attention computations in privacy-preserving settings.

We further observed that, under MPC settings, *gradient-based prompt tuning fails to bring efficiency improvements, and results in slower execution and higher communication overhead.* This is because, while it reduces the number of parameters requiring updates and thereby lowers the computational overhead of privacy-preserving optimization, it fails to avoid the privacy-preserving computations for backward propagation and self-attention mechanisms. Furthermore, compared to model tuning, it incurs additional privacy-preserving forward and backward computations for prompt tokens.

Table 2: Comprehensive performance comparison of SecP-Tuning across various language understanding tasks. The results in the table report the mean and standard deviation over three runs. All experiments are conducted using the pretrained RoBERTa$_{\text{LARGE}}$ model with 16 samples per class.

Method	SST-2 Acc	Yelp P. Acc	AG's News Acc	MRPC F1	RTE Acc	**Avg.**
SFT	$\mathbf{89.86} \pm 1.23$	$\mathbf{93.25} \pm 0.64$	$\mathbf{88.94} \pm 1.12$	$\mathbf{82.15} \pm 3.76$	72.84 ± 4.52	$\mathbf{85.41}$
Prompt Tuning	85.23 ± 1.82	88.47 ± 2.15	85.34 ± 1.32	68.52 ± 4.18	62.53 ± 2.47	78.02
+ Pre-trained prompt	/	/	/	80.35 ± 3.52	$\mathbf{79.80} \pm 1.83$	83.84
FoT	89.56 ± 0.25	91.50 ± 0.16	81.51 ± 0.79	61.56 ± 4.34	52.59 ± 2.21	75.34
+ Pre-trained prompt	/	/	/	75.51 ± 5.54	77.62 ± 1.30	83.14
SecP-Tuning	89.23 ± 0.12	85.30 ± 3.71	79.55 ± 1.32	75.12 ± 3.32	77.32 ± 1.52	82.45

5.3 PERFORMANCE COMPARISON

We evaluated the performance of SecP-Tuning on multiple datasets and compared it with baselines to verify its effectiveness. As shown in Table 2, after utilizing pre-trained prompt embeddings (Gu et al., 2022), SecP-Tuning achieves performance comparable to gradient-based methods, such as SFT and gradient-based prompt tuning. Notably, in simpler sentiment classification tasks, such as SST-2 and Yelp P., SecP-Tuning even outperforms gradient-based prompt tuning. Although the average performance of SecP-Tuning is slightly inferior to gradient-based methods, it offers superior efficiency and deployability, enabling the MPC-based privacy-preserving fine-tuning framework to achieve an optimal balance between *privacy, efficiency, and performance*.

Table 3: We evaluate the feasibility of As-A-Service (AAS), Accuracy, end-to-end time, communication overhead, and the total amount of data uploaded/downloaded for completing PFT on the SST-2 and AG's News datasets.

	AAS	**Acc**	**Fine-tuning Time**	**Communication Volume**	**Upload** (per query)	**Download** (per query)
			SST-2 (Sequence Length: 47)			
SFT	×	87.8	65.86 (h)	67.36 (TB)	-	-
Prompt Tuning	×	72.6	86.15 (h)	149.37 (TB)	-	-
SecP-Tuning	✓	**89.2**	**8.81** (h)	**14.22** (TB)	12 KB	0.5 KB
			AG's News (Sequence Length: 107)			
SFT	×	88.4	75.37 (h)	121.27 (TB)	-	-
Prompt Tuning	×	84.0	80.57 (h)	153.45 (TB)	-	-
SecP-Tuning	✓	**82.1**	**10.43** (h)	**19.68** (TB)	44 KB	2 KB

[3]More advanced MPC operators can further reduce communication overhead and improve fine-tuning speed.

5.4 Deployability Comparison

Beyond efficiency and performance, many other factors must be considered in practical scenarios. As shown in Table 3, we comprehensively compare SecP-Tuning with baseline methods across various dimensions, including serviceability, accuracy, fine-tuning time, communication volume, and the amount of uploaded and downloaded data. To ensure a fair comparison of fine-tuning time, we employ early stopping for all methods: if no improvement in validation accuracy is observed after 1000 steps, the training process is terminated. We find that only SecP-Tuning offers serviceability, allowing data owners to perform PFT directly via APIs provided by the model developer. This ensures that the model developer does not receive any information about the updated parameters. In contrast, gradient-based methods such as SFT and prompt tuning inherently require the model developer to obtain shares of the updated parameters. This introduces the risk of the model developer inferring private fine-tuning data from the updated model parameters. Thus, among all the methods considered, *only SecP-Tuning achieves the best balance in terms of privacy, efficiency, and performance.*

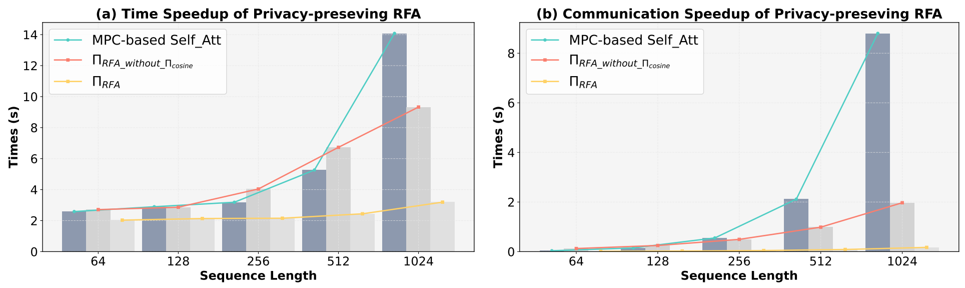

Figure 3: Comparison of Time and Communication Overhead Between Privacy-Preserving RFA and Softmax-Based Privacy-Preserving Self-Attention.

5.5 Comparison of RFA and Self-Attention

We evaluated the privacy-preserving RFA protocol (Π_{RFA}) under varying sequence lengths and compared it with both the MPC-based privacy-preserving self-attention mechanism and the privacy-preserving RFA protocol without the efficient privacy-preserving cosine algorithm proposed in this study ($\Pi_{RFA\_without\_\Pi_{cosine}}$). As illustrated in Figure 3: 1) For the MPC-based privacy-preserving self-attention, Π_{RFA} demonstrates significant improvements in execution speed and communication efficiency. Moreover, as the input length increases, these advantages become increasingly pronounced. This is attributed to the computational complexity of the MPC-based privacy-preserving self-attention mechanism being quadratic with respect to sequence length, whereas the RFA protocol exhibits linear complexity. 2) For $\Pi_{RFA\_without\_\Pi_{cosine}}$, the presence of cosine operations, which are not MPC-friendly, results in relatively limited efficiency gains compared to the MPC-based privacy-preserving self-attention. In fact, for shorter sequence lengths, such as $L = 64$ and $L = 128$, its time and communication overheads even exceed those of the MPC-based privacy-preserving self-attention. This directly highlights that the Π_{cosine} algorithm proposed in SecP-Tuning is the critical factor in enhancing the computational efficiency of privacy-preserving self-attention mechanisms.

6 Conclusion

This paper presents SecP-Tuning, the pioneering MPC-based framework designed for efficient and privacy-preserving prompt tuning of LLMs. By leveraging FoT, it eliminates secure backpropagation and optimizer computations, while introducing a privacy-preserving random feature attention to substitute softmax-based self-attention, thereby circumventing MPC-unfriendly nonlinearities and reducing the computational complexity. Experimental results demonstrate that SecP-Tuning seamlessly integrates efficiency, performance, deployability, and privacy.

Ethics statement

This study focuses on privacy-efficient fine-tuning mechanisms and does not involve ethical or moral concerns. It does not directly collect, generate, or interfere with any personally identifiable information (PII), relying solely on publicly available benchmark datasets (SST-2, MRPC, RTE, Yelp Polarity, AG's News). These datasets are widely used within the research community for English text classification and matching tasks, with licensing terms permitting their use for research purposes. Furthermore, no redistribution of the original data was conducted during the study, and only model performance and efficiency metrics were reported.

Reproducibility statement

This paper provides comprehensive resources to ensure the reproducibility of the experimental results of the proposed SecP-Tuning algorithm. Specifically, a thorough description of the theoretical foundations used in this study, along with relevant references, is included in Section 3. Detailed steps of the proposed methodology are presented in Section 4, and pseudocode for the privacy-preserving algorithms proposed in this paper are provided in Section 6.4. In Section 5.1, we present the experimental setup of SecP-Tuning, including the models, datasets, baseline configurations, dependency libraries, and network environment details. Detailed hyperparameter information is provided in Section 6.6. These elements ensure the reproducibility of the results in this paper. Furthermore, we plan to release the source code upon publication.

References

John Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Advances in neural information processing systems*, 2, 1989.

Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

Zachary Charles, Arun Ganesh, Ryan McKenna, H Brendan McMahan, Nicole Mitchell, Krishna Pillutla, and Keith Rush. Fine-tuning large language models with user-level differential privacy. *arXiv preprint arXiv:2407.07737*, 2024.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, pages 409–437. Springer, 2017.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, 2019.

Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*, 2005.

Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, and Wenguang Cheng. PUMA: Secure inference of LLaMA-7B in five minutes. *arXiv preprint arXiv:2307.12533*, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

C. Dwork and A. Roth. The algorithmic foundations of differential privacy. In *The Algorithmic Foundations of Differential Privacy*, pages 19–20, 2014.

Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM, 1987.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. Ppt: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 8410–8423, 2022.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

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.

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

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. CrypTen: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, 34:4961–4973, 2021.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.243. URL `https://doi.org/10.18653/v1/2021.emnlp-main.243`.

Dacheng Li, Rulin Shao, Hongyi Wang, Han Guo, Eric P Xing, and Hao Zhang. MPCFormer: Fast, performant and private transformer inference with MPC. In *Proceedings of the Eleventh International Conference on Learning Representations, ICLR*, 2023.

Xianzhi Li, Ran Zmigrod, Zhiqiang Ma, Xiaomo Liu, and Xiaodan Zhu. Fine-tuning language models with differential privacy through adaptive noise allocation. *arXiv preprint arXiv:2410.02912*, 2024a.

Yang Li, Wenhan Yu, and Jun Zhao. Privtuner with homomorphic encryption and lora: A p3eft scheme for privacy-preserving parameter-efficient fine-tuning of ai foundation models. *arXiv preprint arXiv:2410.00433*, 2024b.

Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 619–631, 2017.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, 2022.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

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. *Cryptology ePrint Archive*, 2023.

Jinglong Luo, Yehong Zhang, Jiaqi Zhang, Xin Mu, Hui Wang, Yue Yu, and Zenglin Xu. Secformer: Towards fast and accurate privacy-preserving inference for large language models. *arXiv preprint arXiv:2401.00793*, 2024.

Md Jueal Mia and M Hadi Amini. Fedshield-llm: A secure and scalable federated fine-tuned large language model. *arXiv preprint arXiv:2506.05640*, 2025.

Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *Proceedings of 2017 IEEE Symposium on Security and Privacy*, pages 19–38. IEEE, 2017.

Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. BOLT: Privacy-preserving, accurate and efficient inference for transformers. *Cryptology ePrint Archive, Paper 2023/1893*, 2023.

Prajwal Panzade, Daniel Takabi, and Zhipeng Cai. Medblindtuner: Towards privacy-preserving fine-tuning on biomedical images with transformers and fully homomorphic encryption. In *AI for Health Equity and Fairness: Leveraging AI to Address Social Determinants of Health*, pages 197–208. Springer, 2024.

Prajwal Panzade, Javad Rafiei Asl, Daniel Takabi, and Zhipeng Cai. Blindtuner: On enhancement of privacy-preserving fine-tuning of transformers based on homomorphic encryption. *IEEE Internet of Things Journal*, 2025.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, 2007.

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

M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the Asia conference on computer and communications security*, pages 707–721, 2018.

Luis Miguel Rios and Nikolaos V Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuan-Jing Huang, and Xipeng Qiu. BBTv2: Towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3916–3930, 2022a.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pages 20841–20855. PMLR, 2022b.

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

Sameer Wagh, Divya Gupta, and Nishanth Chandran. SecureNN: 3-Party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, pages 26–49, 2019.

Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. *Proceedings on Privacy Enhancing Technologies*, pages 188–208, 2021.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Naiyu Wang, Shen Wang, Meng Li, Longfei Wu, Zijian Zhang, Zhitao Guan, and Liehuang Zhu. Balancing differential privacy and utility: A relevance-based adaptive private fine-tuning framework for language models. *IEEE Transactions on Information Forensics and Security*, 2024.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Guang Yan, Yuhui Zhang, Zimu Guo, Lutan Zhao, Xiaojun Chen, Chen Wang, Wenhao Wang, Dan Meng, and Rui Hou. Comet: Accelerating private inference for large language model by predicting activation sparsity. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 2604–2622. IEEE Computer Society, 2025.

Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Annual Symposium on Foundations of Computer Science*, pages 162–167, 1986.

Wenxuan Zeng, Meng Li, Wenjie Xiong, Wenjie Lu, Jin Tan, Runsheng Wang, and Ru Huang. MPCViT: Searching for MPC-friendly vision transformer with heterogeneous attention. *arXiv preprint arXiv:2211.13955*, 2022.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Yu Zheng, Qizhi Zhang, Sherman SM Chow, Yuxiang Peng, Sijun Tan, Lichun Li, and Shan Yin. Secure softmax/sigmoid for machine-learning computation. In *Proceedings of the 39th Annual Computer Security Applications Conference*, pages 463–476, 2023.

APPENDICES

The appendix is organized as follows.

- In Section 6.1, we report the use of large language models.
- In Section 6.2, We report additional experimental results for SecP-Tuning , including execution time overhead in a wide area network (WAN) environment, performance comparison under different sample sizes, comparison with plaintext performance, and performance on other types of LLMs.
- In Section 6.3, we present the underlying MPC protocols upon which SecP-Tuning is built.
- In Section 6.4, we detail the privacy-preserving algorithms designed for SecP-Tuning, including privacy-preserving cosine similarity and Random Feature Attention (RFA).
- In Section 6.5, we provide a comprehensive security proof of SecP-Tuning.
- In Section 6.6, we report all the hyperparameter settings used in this paper.

6.1 THE USE OF LARGE LANGUAGE MODELS

This work primarily utilized LLMs for academic English translation and refinement. The use of LLMs does not pertain to the significance, innovation, or technical soundness of the core aspects of this work.

Table 4: Efficiency Comparison of RoBERTa$_{\text{LARGE}}$ in WAN Setting. The input sequence length is set to 512, and the number of prompt tokens is set to 50. The results are the average of ten runs.

Bandwidth & Latency	Methods	Forward (s)	Backward (s)	Optimizer (s)	Total (s)
200Mbps/40ms	SFT	605.315	1,718.987	48.036	2,372.338
	Prompt Tuning	847.270	1,997.662	7.810	2,852.742
	SecP-Tuning (FoT)	399.361	0.000	0.133	399.494
	SecP-Tuning (FoT+RFA)	102.923	0.000	0.125	103.048
100Mbps/80ms	SFT	1,502.213	3,893.772	98.822	5,494.807
	Prompt Tuning	2,582.691	4,692.951	10.975	7,286.617
	SecP-Tuning (FoT)	833.448	0.000	0.136	833.584
	SecP-Tuning (FoT+RFA)	211.185	0.000	0.122	211.307

6.2 MORE RESULTS

6.2.1 ADDITIONAL EFFICIENCY RESULTS OF SECP-TUNING

This section presents additional efficiency results of SecP-Tuning. As shown in the data from Table 4, under a WAN setting of 100Mbps/80ms, SecP-Tuning reduces the update time per iteration from 7286.6 seconds in gradient-based Prompt Tuning to 211.3 seconds, achieving approximately 34× acceleration, which significantly surpasses the 16× acceleration observed in a 3Gbps/0.8ms LAN environment. This remarkable improvement stems from its substantial reduction in communication rounds and volume, enabling structural amplification advantages in bandwidth-constrained and high-latency WAN scenarios.

Table 5: Performance comparison under varying number of sample. The results in the table report the mean and standard deviation over three runs. All experiments are conducted using the pretrained RoBERTa$_{\text{LARGE}}$ model.

Mumber of Samples	SFT	Prompt Tuning	FoT
16	85.39 ± 2.84	68.23 ± 3.72	89.56 ± 0.25
32	90.21 ± 2.32	79.32 ± 2.63	90.23 ± 0.31
64	92.17 ± 2.13	87.65 ± 2.55	91.06 ± 0.24
128	$\mathbf{93.26 \pm 2.28}$	92.18 ± 3.57	91.15 ± 0.38

6.2.2 Performance Comparison Across Different Mumber of Samples

In the performance comparison experiments presented in Table 2, FoT demonstrated superior results across multiple tasks compared to gradient-based SFT and Prompt Tuning. Reference [1] hypothesizes that FoT's performance advantage stems from the susceptibility of gradient-based optimization methods to overfitting when dealing with limited training data, whereas FoT, through its exploratory mechanism, often identifies more optimal solutions.

To further investigate performance under larger sample sizes, we conducted additional experiments on the SST-2 dataset, setting the number of samples per class to 16, 32, 64, and 128. As shown in the data from Table 3, when the sample size increases, gradient-based SFT and Prompt Tuning exhibit more robust performance than FoT.

6.2.3 Performance Comparison Across Different Mumber of Samples

In the performance comparison experiments presented in Table 2, FoT demonstrated superior results across multiple tasks compared to gradient-based SFT and Prompt Tuning. Reference [1] hypothesizes that FoT's performance advantage stems from the susceptibility of gradient-based optimization methods to overfitting when dealing with limited training data, whereas FoT, through its exploratory mechanism, often identifies more optimal solutions.

To further investigate performance under larger sample sizes, we conducted additional experiments on the SST-2 dataset, setting the number of samples per class to 16, 32, 64, and 128. As shown in the data from Table 3, when the sample size increases, gradient-based SFT and Prompt Tuning exhibit more robust performance than FoT.

6.2.4 Performance Comparison with Plaintext

In this section, we supplement the comparison experiments between SecP-Tuning and plaintext prompt tuning to analyze the differences in efficiency and performance between the two approaches.

Table 6: Performance comparison with plaintext. The results in the table report the mean and standard deviation over three runs. All experiments are conducted using the pretrained RoBERTa$_{\text{LARGE}}$ model.

Dataset	Method	Fine-Tuning Time	Communication	Accuracy
SST-2	Plaintext	10.1 mins	6.25 KB	89.4
	SecP-Tuning	8.8 hours	14.22 TB	89.2
AG's News	Plaintext	21.0 mins	23 KB	82.6
	SecP-Tuning	10.43 hours	19.68 TB	82.1

The experimental results reveal that SecP-Tuning incurs almost no loss in model utility. This can be attributed to its construction based on cryptographic MPC techniques. As analyzed in Section 6.3 regarding the correctness of MPC protocols, MPC ensures accurate computation results while preserving the privacy of input data. The slight performance degradation may stem from the approximations introduced by MPC for nonlinear activation functions, such as GeLU and LayerNorm. This represents a significant advantage of MPC over other privacy-enhancing methods, such as Differential Privacy (DP) and Federated Learning (FL). However, the drawback lies in the substantial computational and communication overhead it introduces, making it currently challenging to generalize to larger-scale models.

6.2.5 Performance on other types of LLMs

In this section, we include the performance results of SecP-Tuning on other types of LLMs, namely GPT2-LARGE (Radford et al., 2019) and T5-LARGE (Raffel et al., 2020). Experimental results demonstrate that SecP-Tuning is applicable to various architectures of LLMs, including the decoder-only autoregressive model GPT-2 and the encoder-decoder model T5.

Table 7: Performance on other types of LLMs.

Model	DataSet	Accuracy
RoBERTa	SST-2	89.2
	AG's News	82.1
GPT-2	SST-2	75.3
	AG's News	77.7
T5	SST-2	89.1
	AG's News	83.8

6.3 Underlying MPC Protocols

In this section, we provide a brief overview of the underlying protocols used and refer to the works of Knott et al. (2021) and Zheng et al. (2023) for details. Let S_j with $j \in \{0, 1\}$ be two parties that are used to execute the MPC protocol. Each party S_j will be given one additive share $([u]_j, [v]_j) \in \mathcal{Z}_L$ of the operation inputs u and v for $j \in \{0, 1\}$.

6.3.1 Privacy-Preserving Addition, Multiplication and Comparison Protocols

In this section, we provide a detailed description of the execution processes for MPC-based addition, multiplication, and comparison protocols, along with a theoretical analysis of their correctness and privacy guarantees. Other nonlinear privacy-preserving protocols in Section 6.3.2 and Section 6.4.2 can be constructed by invoking these three protocols, and thus their correctness and security can be directly proven based on the aforementioned protocols.

Privacy-preserving addition. Suppose two participants, Alice and Bob, each possess secrets u and v. By executing the addition protocol based on 2-out-of-2 arithmetic secret-sharing ($(2, 2)$-SS), they can compute shares of the output $w = u + v$ while preserving the privacy of inputs u and v. Specifically, the addition protocol based on $(2, 2)$-SS consists of two phases: the secret sharing phase and the computation phase.

In the secret sharing phase:

- Alice locally generates shares of her secret u, i.e., $\mathrm{Shr}(u) \to ([u]_0, [u]_1)$, and sends $[u]_1$ to Bob.
- Bob locally generates shares of his secret v, i.e., $\mathrm{Shr}(v) \to ([v]_0, [v]_1)$, and sends $[v]_1$ to Alice.

In the computation phase:

- Alice computes $[w]_0 = [u]_0 + [v]_0$.
- Bob computes $[w]_1 = [u]_1 + [v]_1$.

Correctness Verification: $[z]_0+[z]_1 = [u]_0+[v]_0+[u]_1+[v]_1 = ([u]_0+[u]_1)+([v]_0+[v]_1) = u+v$.

Privacy Guarantee: During computation, Alice and Bob each possess only one random share of the secrets, ensuring that no information about the original secrets can be inferred.

Privacy-preserving multiplication. Suppose two participants, Alice and Bob, each possess secrets u and v. By executing the multiplication protocol based on 2-out-of-2 arithmetic secret-sharing ($(2, 2)$-SS), they can compute shares of the output $w = u + v$ while preserving the privacy of inputs u and v. Specifically, the addition protocol based on $(2, 2)$-SS consists of two phases: the secret sharing phase and the computation phase.

In the secret sharing phase:

- Alice locally generates shares of her secret x, i.e., $\mathrm{Shr}(x) \to ([x]_0, [x]_1)$, and sends $[x]_1$ to Bob.

- Bob locally generates shares of his secret y, i.e., $\mathrm{Shr}(y) \to ([y]_0, [y]_1)$, and sends $[y]_1$ to Alice.
- Alice possesses the first random shares of the Beaver triple (a, b, c), i.e., $([a]_0, [b]_0, [c]_0)$.
- Bob possesses the second random shares of the Beaver triple (a, b, c), i.e., $([a]_1, [b]_1, [c]_1)$.

- Alice computes $[d]_0 = [x]_0 - [a]_0$ and $[e]_0 = [y]_0 - [b]_0$.
- Bob computes $[d]_1 = [x]_1 - [a]_1$ and $[e]_1 = [y]_1 - [b]_1$.

In the communication phase:

- Alice sends $[d]_0$ and $[e]_0$ to Bob.
- Bob sends $[d]_1$ and $[e]_1$ to Alice.

In the computation phase:

- Alice reconstructs $d = [d]_0 + [d]_1 = x - a$ and $e = [e]_0 + [e]_1 = y - b$.
- Bob reconstructs d and e similarly.
- Alice computes $[z]_0 = [x]_0 e + d[y]_0 + [c]_0$.
- Bob computes $[z]_1 = -de + [x]_1 e + d[y]_1 + [c]_1$.

Correctness Verification:

$$\begin{aligned}
[z]_0 + [z]_1 &= [x]_0 e + d[y]_0 + [c]_0 - de + [x]_1 e + d[y]_1 + [c]_1 \\
&= ([x]_0 + [x]_1)e + ([y]_0 + [y]_1)d - de + c \\
&= x(y-b) + y(x-a) - (x-a)(y-b) + c \\
&= xy - xb + xy - ay - xy + ay + xb - ab + c \\
&= xy.
\end{aligned}$$

Privacy Guarantee: During computation, Alice and Bob possess only one random share each of a and b. Since a and b are randomly generated and independent of the inputs x and y, no information about x or y is revealed.

Privacy-preserving comparison. Similarly, Alice holds secret u and Bob holds secret v, and the comparison can be implemented as follows:

- Alice and Bob first generate the shares of their respective private inputs, a.k.a., $[u]$ and $[v]$, as privacy-preserving addition.
- Two parties locally compute $[w] = [u] - [v]$.
- Two parties jointly invoke the Arithmetic-to-Boolean conversion (Knott et al., 2021) to convert $[w]$ from Arithmetic sharing to Boolean sharing $\langle z \rangle = \mathsf{A2B}([w])$.
- Two parties locally extract the most significant bit of Boolean sharing $\langle z \rangle$ as $\langle b \rangle = \langle w \rangle \gg (\ell - 1)$[4].
- Finally, the additive shares of $[u < v]$ can be derived by converting Boolean sharing $\langle b \rangle$ to Arithmetic sharing $[b]$ using Boolean-to-Arithmetic conversion protocol (Knott et al., 2021).

Correctness & Privacy. Except for sharing the inputs, the computation phase consists of $\log_2 \ell + 1$ rounds of communication and transmits 3456 bits. The correctness is easy to follow, and the privacy guarantee is inherent from well-established 2PC basic primitives.

6.3.2 Privacy-Preserving Non-Linear Protocols

Privacy-preserving maximum. The maximum of the n-element vector $\boldsymbol{x}$ is implemented by calling $\log_2 n$ privacy-preserving comparisons using the tree reduction algorithm (Knott et al., 2021).

[4] $\gg \ell$ denotes shift ℓ bits to the right.

Privacy-preserving exponential. The exponential function is complex and usually implemented using the repeated-squaring approximation method

$$e^x = \lim_{x\to\infty}\big(1+\frac{x}{2^n}\big)^{2^n}, \tag{7}$$

which converts exponential calculations into addition and square operations. By fault, iterations are set $n = 8$ in (Knott et al., 2021).

Privacy-preserving reciprocal. Function reciprocal $\frac{1}{x}$ is implemented by the Newton-Raphson method, which converts reciprocal calculations into addition and multiplication operations. The iterative formula is

$$y_{n+1} = y_n(2 - xy_n). \tag{8}$$

The initial value of the iteration is

$$y_0 = 3e^{\frac{1}{2}-x} + 0.003. \tag{9}$$

The number of iterations is set to 10 in (Knott et al., 2021) by default.

Privacy-preserving square root. $\sqrt{x}$ is approximated by the Newton-Raphson method in MPC, which converts exponential calculations into addition and multiplication operations. The iterative formula is

$$y_{n+1} = \frac{1}{2}y_n(3 - xy_n^2). \tag{10}$$

The initial value of the iteration is

$$y_0 = e^{-2.2(\frac{x}{2}+0.2)} + 0.198046875. \tag{11}$$

The number of iterations is set to 3 in (Knott et al., 2021) by default.

6.4 Privacy-preserving Protocols

6.4.1 Privacy-preserving Cosine

We propose an efficient privacy-preserving cosine protocol Π_{cosine} by exploiting the periodicity of the cosine function and trigonometric addition identity formulas. Here's a detailed description of the algorithm steps: In the offline phase, the protocol initiates by generating pseudo-random values. Specifically, S_0 and the trusted third party T jointly produce $[t]_0, [u]_0, [v]_0$ by evaluating a pseudo-random function (PRF) with a specific key k_0. Similarly, S_1 and T generate $[t]_1$ using a different key k_1. Then, the trusted third party T recover the actual value $t = [t]_0 + [t]_1$, calculates $[u]_1 = \sin(t) - [u]_0$ and $[v]_1 = \cos(t) - [v]_0$. This phase is crucial for preparing necessary correlated randomness that will be used in the online phase.

In the online phase, the parties compute the $[\sin(x)]$ securely. First, each party S_j computes $[\delta]_j = [x]_j + [t]_j \pmod{\tau}$, where τ represents the periodicity of the trigonometric function, such as 20. Then, through one round of communication, the parties reconstruct δ by exchanging $[\delta]_0$ and $[\delta]_1$. Subsequently, we get $p = \sin(\delta)$ and $q = \cos(\delta)$. Finally, each party calculates $[y]_j = p[u]_j + q[v]_j$. This effectively leverages the precomputed correlated randomness with the current input $[x]$ to produce the $[\sin(x)]$ in a privacy-preserving manner. The Π_{cosine} requires only one round of communication during the online phase, with a communication cost of transmitting 2ℓ elements.

Correctness Verification:

$$\begin{aligned}
[y]_0 + [y]_1 &= p[u]_0 + q[v]_0 + p[u]_1 + q[v]_1 \\
&= p([u]_0 + [u]_1) + q([v]_0 + [v]_1) \\
&= \sin(\delta)\sin(t) + \cos(\delta)\cos(t) \\
&= cos(\delta - t) \\
&= cos(x + t - t) \\
&= cos(x).
\end{aligned}$$

Privacy Guarantee: During the computation process, the server S_j only obtains the information of $[x]_j$, $[t]_j$, $[\delta]_j$, and δ. Since $\delta = x + t \pmod{\tau}$ and t is independent of x, δ is also independent of x. Therefore, S_j cannot gain any information about the private input x during execution.

Algorithm 1: Protocol for Privacy-preserving Cosine Π_{cosine}

```
Input: For $j \in \{0,1\}$, $S_j$ holds the shares $[x]_j$; Pseudo-Random Function (PRF), and key $k_j$.
Output: For $j \in \{0,1\}$, $S_j$ returns the shares $[y]_j$, where $y = \cos(x)$.
/* Offline Phase */
1 $S_0, T : [t]_0, [u]_0, [v]_0 \leftarrow PRF(k_0)$
2 $S_1, T : [t]_1 \leftarrow PRF(k_1)$
3 $T : t = [t]_0 + [t]_1, [u]_1 = \sin(t) - [u]_0, [v]_1 = \cos(t) - [v]_0$
/* Online Phase */
4 $[\delta]_j = [x]_j + [t]_j \pmod{\tau}$
5 $\delta = [\delta]_0 + [\delta]_1$ // reconstruct $\delta$ by 1 round of communication
6 $p = \sin(\delta), q = \cos(\delta)$
7 $[y]_j = p[u]_j + q[v]_j$
```

6.4.2 Privacy-preserving Feature Attention

The Privacy-preserving RFA Protocol (Π_{RFA}) is designed to enable computation of RFA with privacy preservation. The algorithm involves two parties, S_0 and S_1, and a trusted third party T, to collaboratively compute the RFA while keeping the input data secure. In the offline phase, the protocol begins with the generation of pseudo-random values. Specifically, S_0 and the trusted third party T jointly produce $[t]_0, [u]_0, [v]_0$ by evaluating a PRF with a random seed r_0, and also generate matrix W using another random seed r. On the other hand, S_1 and the trusted third party T generate $[t]_1$ by evaluating the PRF with a different random seed r_1, and use the same matrix W generated earlier. Then, the trusted third party T recovers the actual value $t = [t]_0 + [t]_1$. Based on t, T computes $[u]_1 = \sin(t) - [u]_0$ and $[v]_1 = \cos(t) - [v]_0$. This offline phase essentially prepares some necessary random values and parameters, which will be used in the online phase. Although these values are related to trigonometric functions, they are computed in a way that preserves privacy as the actual values are hidden within the shares.

In the online phase, the algorithm focuses on computing the attention mechanism. First, for each query q_t at time step t and key k_i, the corresponding feature mappings are computed. This is done by taking the shares of q_t and k_i (i.e., $[q]_t$ and $[k]_i$) and applying a cosine-based transformation denoted as Π_{cosine}, scaled by a factor of $\sqrt{2/r}$. The scaling factor is important to ensure proper normalization of the feature mappings.

Next, for each key-value pair (k_i, v_i), the share $[z]_j$ is computed as the element-wise product (denoted by $\otimes$) between the feature - mapped key $[\phi(k_i)]_j$ and the value v_i. This effectively combines the key's feature representation with its associated value.

Then, the attention score $[s]_j$ is calculated as the dot product between the feature-mapped query $[\phi(q_t)]_j$ and the feature-mapped key $[\phi(k_i)]_j$. This dot product represents the similarity between the query and the key in the transformed feature space.

Finally, the output share $[y]_j$ is obtained by dividing $[z]_j$ by $[s]_j$. This step normalizes the combined key-value representation by the attention score, resulting in the weighted value that will be used as the output of the attention mechanism. The division here is crucial as it implements the attention-weighting process, where the value is scaled according to how relevant the corresponding key is to the query.

6.5 Security Analysis

SecP-Tuningadheres to a semi-honest (also known as honest-but-curious) assumption similar to the works of Li et al. (2023) and Dong et al. (2023), where honest participants constitute the majority. Under this assumption, the security of SecP-Tuningcan be formally proved against static semi-honest adversaries denoted as $\mathcal{A}$, which can potentially corrupt no more than one of the servers in the hybrid model.

SecP-Tuningis constructed from the well-established sub-protocols of Knott et al. (2021); Zheng et al. (2023), and we invoke these protocols in a black-box manner. Leveraging the concept of

Algorithm 2: Privacy-preserving RFA Protocol (Π_{RFA})

```
Input: For $j \in \{0,1\}$, $S_j$ holds the shares $\{[q]_t, [k]_i, [v]_i\}$;
Output: For $j \in \{0,1\}$, $S_j$ returns the shares $[y]_j$, where $y = RFA([q]_t, [k]_i, [v]_i)$.
/* Offline Phase */
1 $S_0, T : [t]_0, [u]_0, [v]_0 \leftarrow PRF(r_0); W \leftarrow PRF(r)$
2 $S_1, T : [t]_1 \leftarrow PRF(r_1); W \leftarrow PRF(r)$
3 $T : t = [t]_0 + [t]_1, [u]_1 = \sin(t) - [u]_0, [v]_1 = \cos(t) - [v]_0$
/* Online Phase */
4 $[\phi(q_t)]_j = \sqrt{\frac{2}{r}}\Pi_{cosine}(W[q_t]_j)$; $[\phi(k_i)]_j = \sqrt{\frac{2}{r}}\Pi_{cosine}(W[k_i]_j)$
5 $[z]_j = [\phi(k_i)]_j \otimes v_i$
6 $[s]_j = [\phi(q_t)]_j \cdot [\phi(k_i)]_j$
7 $[y]_j = [z]_j / [s]_j$
```

composable security established by Canetti (2001), it is easy to see that the security of SecP-Tuningis guaranteed in the sub-protocols hybrid model.

Table 8: Core and auxiliary hyper-parameters for Feedforward-only Tuning (FoT) and Random Feature Attention (RFA).

Name	**Symbol**	**Default**	**Description**
Batch size	-	16	-
Optimizer	-	CMA-ES	Derivative-free evolutionary strategy (no backward propagation required).
Prompt length	L	50	Number of continuous prompt tokens (controls raw dimension $D = L \times d_{\text{emb}}$).
Initial prompt	p_0	NLI-pretrained	Pretrained prompt (e.g., on MNLI) for sentence-pair tasks.
Subspace dimension	d	500	Dimension of the low-dimensional search subspace; trade-off between coverage and GFO efficiency.
Population size	λ	20	Number of CMA-ES offspring per generation (heuristic: $4 + 3\log d$).
Random projection	A	Uniform	Projection matrix $A \in \mathbb{R}^{D\times d}$ sampled from a uniform distribution (found superior to Gaussian).
Loss function	$L(\cdot)$	Cross-Entropy	Provides dense supervisory signal under few-shot regime.
API call budget	-	8000	Maximum number of model inference calls (evaluation points).
Early stopping	-	1000	Stop if dev accuracy shows no improvement for 1000 evaluations.
Kernel type	-	Gaussian	Shift-invariant kernel approximated by random features.
Number of Random features	D	128	Number of random features per head.
Random feature vectors	$\{w_i\}$	$w_i \sim \mathcal{N}(0, I)$	Base Gaussian samples; drawn once (fixed) for reproducibility.

6.6 Hyperparameters

6.6.1 Hyperparameters of Gradient-based Fine-Tuning

For gradient-based fine-tuning methods, including SFT and gradient-based Prompt Tuning, we utilized the standard Adam optimizer with a batch size of 16. Additionally, we explored a wider range of learning rates and implemented an early stopping mechanism to mitigate overfitting. Specifically, for SFT, the learning rates were selected from the range [1e-6, 3e-6, 5e-6, 1e-5, 3e-5, 5e-5, 1e-4], with a maximum of 200 epochs and an early stopping patience of 30 steps, meaning training would terminate if no improvement was observed on the validation set for 30 consecutive steps. For gradient-based Prompt Tuning, the learning rates were chosen from the range [1e-5, 3e-5, 5e-5, 1e-4, 3e-4, 5e-5, 1e-3], with a maximum of 1000 epochs and an early stopping patience of 50 steps, indicating that training would halt if no improvement was observed on the validation set for 50 consecutive steps.

6.6.2 Hyperparameters of Feedforward-only Tuning and Random Feature Attention

To ensure the reproducibility of the experimental results, SecP-Tuning adopted the same hyperparameter settings as those used by its forward propagation plugin and random feature attention mechanism plugin. The specific hyperparameter names, symbols, values, and descriptions are detailed in Table 8.