ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240958

SecPE: Secure Prompt Ensembling for Private and Robust Large Language Models

Jiawen Zhang[a,1], **Kejia Chen**[a,1], **Zunlei Feng**[a,*], **Jian Lou**[a,**] **and Mingli Song**[a]

[a]State Key Laboratory of Blockchain and Data Security, Zhejiang University
[b]Hangzhou High-Tech Zone (Binjiang) Blockchain and Data Security Research Institute

Abstract. With the growing popularity of LLMs among the general public users, privacy-preserving and adversarial robustness have become two pressing demands for LLM-based services, which have largely been pursued separately but rarely jointly. In this paper, to the best of our knowledge, we are among the first attempts towards robust and private LLM inference by tightly integrating two disconnected fields: private inference and prompt ensembling. The former protects users' privacy by encrypting inference data transmitted and processed by LLMs, while the latter enhances adversarial robustness by yielding an aggregated output from multiple prompted LLM responses. Although widely recognized as effective individually, private inference for prompt ensembling together entails new challenges that render the naive combination of existing techniques inefficient.

To overcome the hurdles, we propose SecPE, which designs efficient fully homomorphic encryption (FHE) counterparts for the core algorithmic building blocks of prompt ensembling. We conduct extensive experiments on 8 tasks to evaluate the accuracy, robustness, and efficiency of SecPE. The results show that SecPE maintains high clean accuracy and offers better robustness at the expense of merely 2.5% efficiency overhead compared to baseline private inference methods, indicating a satisfactory "accuracy-robustness-efficiency" tradeoff. For the efficiency of the encrypted ARGMAX operation that incurs major slowdown for prompt ensembling, SecPE is 35.4 times faster than the state-of-the-art peers, which can be of independent interest beyond this work.

1 Introduction

Large language models (LLMs) have garnered a meteoric rise in popularity among general public users due to their remarkable performance across myriad natural language processing (NLP) tasks [36, 38]. LLMs are oftentimes deployed by service providers in the form of Machine Learning as a Service (MLaaS) [39, 23], whereby users can conveniently exploit the full potential of LLM by submitting their inference data, prepended by specific prompts from prompt learning techniques [18], to obtain high-performing LLM outputs tailored to their downstream tasks. Accompanying this widespread adoption, there arise privacy and robustness concerns for LLMs [13].

Privacy concerns and private inference. On the privacy aspect, users' inference data can inadvertently reveal sensitive information if transmitted and processed by the LLM service provider in plaintext [39, 23], risking identification and privacy breaches. Additionally, the user-submitted prompts can be valuable intellectual property and also raise privacy concerns. As a result, both inference data and user-side prompts demand privacy-preserving measures [13]. Among the many attempts to avoid submitting raw data for LLM inference, private inference offers very strict privacy protection by allowing inference to be conducted on encrypted data. For instance, Fully Homomorphic Encryption (FHE) allows rich computations (covering most operations needed in LLM inference) on encrypted data without exposing sensitive information [9]. By encrypting inputs using FHE, only encrypted predictions are sent to the server, ensuring privacy throughout the process. As legal and societal pressures mount, service providers' adoption of such privacy-preserving technologies has received increasing research attention.

Robustness concern and prompt ensembling. On the robustness aspect, it is well-recognized that the output of LLMs can be manipulated by subtle yet deliberate changes in the inference sample or the prompt [33]. There has been a growing focus on enhancing the robustness of LLMs, especially in safety-critical downstream application areas. Various methods have been proposed, ranging from more advanced (and sophisticated) to simple methods [7]. One representative method from the latter category follows the idea of prompt ensembling [25], which involves making multiple inferences for a single inference data and providing the aggregated result as the final prediction.

This study. The current research efforts on safeguarding privacy and robustness during LLM inference are largely explored separately. Driven by the simultaneous demands from both privacy and robustness aspects, we envision that these two aspects should be pursued jointly. Among the first attempts toward mitigating both concerns of LLMs jointly, we investigate the potential to achieve private and robust LLM inference through tight integration of private inference and prompt ensemble. We focus on these two techniques due to their effectiveness in addressing their respective concerns. In particular, we note that while there may be more advanced techniques for enhancing robustness than prompt ensembling, achieving a balance between robustness and efficiency within the private inference workflow of the simpler prompt ensembling method already poses significant challenges. That is, naive application of existing private inference methods for prompt ensembling entails great efficiency overhead. The crux of efficient private inference for prompt ensembling is that the aggregation operation introduced by prompt ensembling, albeit simple and efficient in plaintext computation, requires pro-

* Corresponding Author. Email: zunleifeng@zju.edu.cn
** Corresponding Author. Email: jian.lou@zju.edu.cn
[1] Equal contribution.

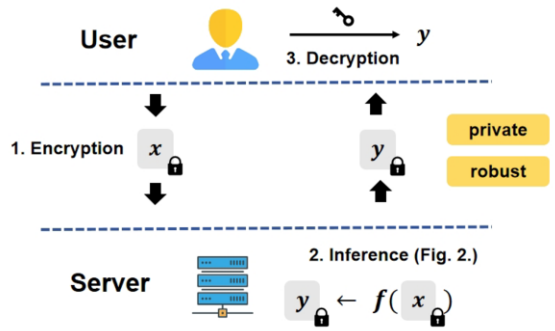

Figure 1. A high-level overview of SecPE for private and robust LLM inference in FHE-based MLaaS.

hibitive computation in the ciphertext.

To overcome the inefficiency challenges, we propose SecPE: a new secure prompt ensembling method for private and robust LLM inference. As illustrated in Figure 1, SecPE allows user to encrypt their inference data and prompts before transmitting them to the LLM server for inference. The inference results from the LLM server are aggregated from multiple prompted responses and transmitted back to the user in ciphertext format, which can be decrypted only by the user's private key. The encrypted aggregation operation heavily relies on efficient computation of ARGMAX, which is unfortunately not readily supported by the common homomorphic primitives like the RNS-CKKS FHE scheme [16]. Lying at the design core of SecPE is a new efficient private aggregation algorithm to be presented in Algorithm 1, which resorts to an efficient approximation of ARGMAX to circumvent this efficiency bottleneck. We conduct extensive experiments to test the accuracy, robustness, and efficiency of SecPE across 6 tasks from GLUE/AdvGLUE and 2 tasks from mathematical reasoning data sets. We also extend our SecPE to Text-Image Models and test the accuracy across 7 tasks from CIFAR, ImageNet, and their variants. The results show that SecPE is capable of maintaining both high utility and robustness while providing privacy protection.

The main contributions of this paper are summarized as follows:

- To the best of our knowledge, we are among the first to jointly study the privacy and robustness concerns of LLM inference, which become increasingly pressing considering the growing deployment of LLM-based services.
- We propose SecPE to achieve private and robust LLM inference, which devises new secure primitives tailor-made for prompt ensembling to strike a satisfactory "accuracy-robustness-efficiency" tradeoff.
- We conduct extensive experiments on 15 tasks from 4 popular benchmarks to corroborate the superior performance of SecPE against baseline methods.

2 Background

2.1 Privacy Issues of LLMs

LLMs such as the GPT have revolutionized natural language processing and understanding with human-level proficiency [14, 3]. However, with their increasing deployment in MLaaS by service providers and growing popularity among the general public users, there arise aggravating privacy concerns. In the typical MLaaS serving setting, users submit inference data to the remote server hosting a proprietary model and receive predictions in return. Users therefore have privacy concerns about their inference data that, despite being sensitive or even confidential, are transmitted and processed in plaintext by the MLaaS service provider [28]. This issue has even led to ChatGPT being temporarily banned in Italy [19, 20]. Recognizing this pressing privacy concern, existing works introduce various means to avoid direct transmission and processing inference data in plain text form.

Private inference emerges as a viable solution, promising to reconcile the need for high-performant inference data processing with strict privacy requirements [30, 11, 21]. Private inference provides a way to guarantee the privacy and confidentiality of both the inference data and the proprietary LLM. It ensures that data is not transmitted or processed in plaintext but as ciphertext, thereby safeguarding sensitive details about the server's model weights and the user's inputs from disclosure. While private inference has significant applications in computer vision and image processing [41], its use in LLMs is nascent. Notably, the integration of private inference in prompt learning settings and prompt ensembles remains an under-explored area, presenting a frontier yet to be ventured into the field.

By pursuing private inference tailored for prompt ensemble learning, we aim to bridge the gap between utility, robustness, and privacy, thereby realizing the benefits of prompted LLMs without compromising user trust and data integrity.

2.2 Fully Homomorphic Encryption

The FHE scheme used in this paper is the full *residue number system* (RNS) variant of Cheon-Kim-Kim-Song (CKKS) [5]. RNS-CKKS is a *leveled* FHE, which can support computations up to a multiplicative depth L. Both the plaintexts and ciphertexts of RNS-CKKS are elements in a polynomial ring:

$$\mathcal{R}_Q = \mathbb{Z}_Q[X]/(X^N + 1)$$

where $Q = \Pi_{i=0}^{L} q_i$ with distinct primes q_i. Once a ciphertext's level becomes too low, a *bootstrapping* operation is required to refresh it to a higher level, enabling more computations. In a nutshell, bootstrapping homomorphically evaluates the decryption circuit and raises the modulus from q_0 to q_L by leveraging the isomorphism $\mathcal{R}_{q_0} \cong \mathcal{R}_{q_0} \times \mathcal{R}_{q_1} \times \cdots \times \mathcal{R}_{q_L}$ [2]. Suppose the bootstrapping consumes K levels, then a fresh ciphertext can support $L - K$ levels of computations.

RNS-CKKS supports *single instruction multiple data* (SIMD), which enables encrypting a vector with N elements into a single ciphertext and processing these encrypted elements in a batch without introducing any extra cost. Below, we summarize the homomorphic operations used in this paper:

- $a \oplus b$. The addition takes two SIMD ciphertexts a and b; outputs $[a_0 + b_0, a_1 + b_1 ..., a_{N-1} + b_{N-1}]$.
- $a \ominus b$. The subtraction takes two SIMD ciphertexts a and b; outputs $[a_0 - b_0, a_1 - b_1 ..., a_{N-1} - b_{N-1}]$.
- $a \otimes b$. The multiplication takes two SIMD ciphertexts a and b; outputs $[a_0 \times b_0, a_1 \times b_1 ..., a_{N-1} \times b_{N-1}]$.
- $RotL(a, s)$. The left-rotation takes one SIMD ciphertext a and an integer s; left-rotates the vector by s slots.
- $RotR(a, s)$. The right-rotation takes one SIMD ciphertext a and an integer s; right-rotates the vector by s slots.

2.3 Prompt Ensembling for Robust LLMs

The brittleness of LLMs to slight input modifications often leads to varied/inaccurate and sometimes even malicious/harmful out-

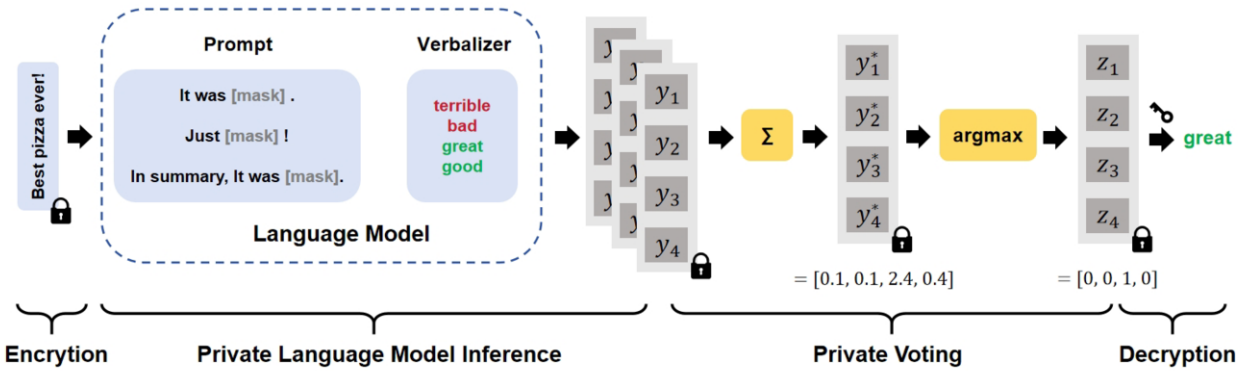

Figure 2. An illustration of SECPE, which enables homomorphically encrypted LLM inference with guarantees.

puts, highlighting the essential need for enhanced robustness for LLMs [27]. Robustness in this context refers to LLM's ability to provide consistent predictions regardless of slight changes to the inference data, aiming for more predictable and stable responses.

Building on the success of prompt learning, prompt ensemble learning [1] demonstrates the potential to offer efficient, effective, and robust predictions. Prompt ensemble utilizes a series of prompts to allow for the aggregation of multiple responses for the same inference data, leading to more robust predictions.

Prompt ensembling, in which the masked language model $\mathcal{L}$ is directly tasked with "auto-completing" natural language prompts. For instance, for the inference data x_{in}, the template into which the inference data is inserted that x_{prompt} = "It was `MASK`" is concatenated (i.e., $x_i = x_{in} \oplus x_{\text{prompt}}$), The prompt typically includes one or more masked tokens `[MASK]` that the model $\mathcal{L}$ is expected to fill in, making it a structured query that directs the model's response.

The single output refers to the model's prediction for each prompt, drawing on the context of the prompt and input data present, like determining the sentiment of a movie review. When multiple prompts or input variations are used to obtain a range of model responses, the aggregated output synthesizes these individual outputs to derive a more robust or accurate prediction. This aggregation could involve combining the model's responses to enhance prediction reliability or accuracy, especially in tasks where nuanced understanding or multiple aspects of the input data are considered.

For NLP tasks, suppose there are m prompt templates, the verifier takes a question and a candidate reasoning path as input and outputs the probability that the reasoning path leads to the correct answer [17].

$$y^* = \text{ARGMAX}(\sum_{i=1}^{m} f(x_{in} \oplus x_{\text{prompt}_i}))$$

where $f(\cdot)$ is the probability produced by the verifier's model $\mathcal{L}$.

In addition to LLM, prompt ensembling is also widely used in text-image models. For zero-shot image classification tasks, suppose there are m prompt templates, prompt ensembling as proposed in CLIP [22] generalizes:

$$y^* = \text{ARGMAX}(\sum_{i=1}^{m} (I(x_{in}) \cdot T(x_{\text{prompt}_i})))$$

where I is the image encoder and T is a text encoder.

3 Proposed Method: SecPE

We propose a new private inference framework tailor-made for the prompt ensembling. Private inference for prompt ensembling raises a critical, unaddressed issue: the challenge of integrating private contextual inference. Incorporating privacy-preserving mechanisms into prompt ensembles remains a significant and complex challenge, despite progress in leveraging prompt-based learning to improve model effectiveness in downstream tasks. Our work aims to break new ground by developing a comprehensive framework that not only improves model performance through optimized prompt selection but also prioritizes the integration of robust privacy safeguards.

3.1 SecPE *Framework*

We give an illustration of SecPE in Fig 2, the overall process is divided into the following four steps:

1. **Encryption.** User encrypts m inputs $x_i = x_{in} \oplus x_{\text{prompt}}, i \in [1, m]$ using FHE and sends them to the server, where m is the number of prompt templates.
2. **Private Language Model Inference.** Server uses the language model $\mathcal{L}$ classifying m inputs into one of n classes, n is the number of labels. the inputs are propagated through $\mathcal{L}$ utilizing the homomorphic operations of the FHE scheme to obtain m encrypted logits $y_i, i \in [1, m]$.
3. **Private Voting.** Server aggregates the encrypted logits $y^* \leftarrow \sum_{i=1}^{m} y_i$ and then evaluates ARGMAX function in FHE. In particular, this step transforms the logit vector y^* into a one-hot vector z. Then the server sends z to the User.
4. **Decryption.** User decrypts z with its secret key, where the single non-zero entry represents the index of the predicted classification label.

As illustrated in the preceding workflow of SecPE, Steps 1 and 4 pertain to fundamental FHE encryption and decryption operations. Step 2 has been implemented across numerous recent works, including in [11, 21]. These three steps are orthogonal to the efficiency designs of prompt ensembling. The primary obstacle lies in leveraging FHE to access the ARGMAX operation in Step 3.

It is important to note that private voting can only be calculated by the server in ciphertext and cannot be handed over to the user in plaintext. This is because numerous works, including those by[29, 40, 37], have designed membership inference attacks based on the

class probability distribution of the prediction vector. It is for this reason that the output of the final layer, commonly referred to as the logits, is generally considered to represent the raw confidence ratings associated with the predictions. These ratings are selected using the ARGMAX processing, whereby the one with the highest probability is selected from the available ratings. It is important to note that the simple act of returning these logits without the ARGMAX process carries the risk of exposing more information about the underlying data and the decisions made by the model, potentially resulting in privacy leakage.

The fact that FHE does not permit control flow evaluation (e.g., branching) and that ciphertext comparison (e.g., inequality checking) is not directly supported by the homomorphic primitives of the RNS-CKKS FHE scheme means that we cannot implement the ARGMAX algorithm in a canonical manner. Instead, we are seeking an efficient approximation to circumvent the efficiency bottleneck that is introduced by prompt ensembling.

3.2 Efficient Private Inference for Prompt Ensembling

As mentioned above, the design core of efficient private inference for prompt ensembling lies at the private aggregation operator, i.e., the ARGMAX operation.

Therefore, our goal is to approximate the following function on an RNS-CKKS ciphertext logit vector:

$$[y_1, ..., y_n, 0^{N-n}] \rightarrow [z_1, ..., z_n, \#^{N-n}], \tag{1}$$

where $z_i = 1$ for the index i corresponding to the largest value among $[y_1, y_2, ..., y_n]$ (and 0 elsewhere).

The state-of-the-art non-interactive protocol that can achieve this goal is Phoneix [12]. Phoenix adopts the idea of bubble sorting to compare each element with adjacent elements by rotating the ciphertext and making a difference with the input:

$$\begin{aligned} \mathbf{s}_1 &\leftarrow Sign(\mathbf{y} - RotL(\mathbf{y}, 1)) \\ \mathbf{s}_2 &\leftarrow Sign(\mathbf{y} - RotL(\mathbf{y}, 2)) \\ &... \\ \mathbf{s}_m &\leftarrow Sign(\mathbf{y} - RotL(\mathbf{y}, m)) \end{aligned}$$

So $\mathbf{s} \leftarrow \sum_{i=1}^{m} \mathbf{s}_i$ counts the comparison result among each input and adjacent elements. Obviously, the value of the maximum element position is m, and the values of other positions are less than m. After that, through simple linear transformation, $\mathbf{z}$ can be obtained based on $\mathbf{s}$ (cf. Phoenix [12] for details).

However, this method requires $(m+1)$ times $Sign$ operations and $(m+1)$ times ciphertext rotations, which is very inefficient when m is large (e.g. $m = 1024$ in CLIP [22]). To solve the problem, we innovatively proposed an ARGMAX evaluation method as:

$$z_i \leftarrow Sign(y_i - y_{max}) + 1. \tag{2}$$

To enable encrypted comparisons, we leverage the polynomial approximation of the sign function:

$$Sign(x) = \begin{cases} -1 & -1 \le x \le -2^{-\alpha} \\ 0 & x = 0 \\ 1 & 2^{-\alpha} \le x \le 1 \end{cases} \tag{3}$$

The approximation involves a composition of polynomials:

$$Sign(x) = f^{d_f}(g^{d_g}(x)) \tag{4}$$

Algorithm 1 ARGMAX on RNS-CKKS

Input: $[y_1, y_2, ..., y_n, 0^{N-n}]$
Output: $[z_1, z_2, ..., z_n, \#^{N-n}])$ as in Eq. 1

1: **function** ARGMAX(y)
2: $y \leftarrow y \oplus RotR(y, n)$
3: $y_{max} \leftarrow QuickMax(y)$
4: $y \leftarrow y \ominus y_{max}$
5: $z \leftarrow Sign(y)$
6: $z \leftarrow z \oplus 1$
7: **return** z
8: **end function**
9: **function** $QuickMax(y)$
10: $l \leftarrow \log_2 n$
11: **for** $i = 0$ to $\log n - 1$ **do**
12: $r \leftarrow RotL(y, 2^i)$
13: $r \leftarrow Max(r, y)$
14: $y \leftarrow r$
15: **end for**
16: **return** y
17: **end function**

where $f()$, $g()$ are two polynomials and d_f, d_g are the number of repetitions for them. In our implementation, both $f()$ and $g()$ are 9-degree polynomials; we set $\alpha = 12, d_f = 2, d_g = 2$, so the max error bound is less than 10^{-4}. To reduce the multiplicative depth, we evaluate the polynomials using the Baby-Step-Giant-Step algorithm [10].

Before proceeding, we comment on the basic input requirement of $Sign(x)$, namely that its inputs are in $[-1, 1]$. Suppose the inputs $x_i \in [D_{min}, D_{max}]$, to ensure this requirement, for those inputs that need to be different from each other, we need to normalize $\hat{x_i} \in [0, 1]$:

$$\hat{x_i} = \frac{x_i - D_{min}}{D_{max} - D_{min}}, \tag{5}$$

meaning that for all $i \neq j$, $\hat{x_i} - \hat{x_j} \in [-1, 1]$, satisfying the requirement in Algo.1

In order to get x_{max}, with the help of the $Sign$ function, we can calculate the maximum value of a and b by:

$$Max(a, b) = \frac{a+b}{2} + \frac{a-b}{2} \cdot Sign(a-b). \tag{6}$$

Then, the selection vector can be easily computed as described in Algorithm 1.

In Fig. 3, we illustrate how Alg. 1 processes a toy example. The algorithm first duplicates the logits (Line 2), then uses $QuickMax$ to get the maximum value of $[y_1, y_2, ..., y_n]$. Unlike phoenix [12], we do not rotate only one step at a time, but rotate 2^i, $i \in [0, \log n - 1]$ steps each time, which greatly reduces our number of rotations and the number of $Sign$ operations. This technique can be applied to all associative operations, such as sum, maximum or minimum, etc.

Note that we need to use $2n$ slots to calculate the ARGMAX of an input of length n. Since the total number of slots of the ciphertext polynomial is $N \gg n$, we can batch process the ARGMAX of $\frac{N}{2n}$ inputs in parallel in a polynomial ciphertext. For example, if the SIMD slot is 32768, and the input length is 256, we can batch 64 inputs in parallel.

Method	Prompt	Setting	SST-2	QQP	MNLI-m	MNLI-mm	RTE	QNLI
LM-BFF (Plaintext)	Single	Cln	94.0	80.1	76.7	78.3	78.1	81.4
		Adv	54.1	46.2	47.1	40.1	58.8	61.5
LM-BFF (Ciphertext)	Single	Cln	93.7	79.2	76.0	77.6	77.5	81.0
		Adv	53.8	46.1	46.4	39.5	58.2	61.1
PET (Plaintext)	Ensemble	Cln	93.4	73.7	74.6	75.7	74.2	84.6
		Adv	61.7	59.3	55.6	44.8	54.0	67.9
SecPE	Ensemble	Cln	93.0	73.1	73.2	74.7	72.2	81.1
		Adv	61.3	59.3	55.4	43.9	53.2	66.8

Table 1. Performance comparison on GLUE (Cln) and Adversarial GLUE (Adv) benchmarks. We report the average and standard deviation in the accuracy values of 5 different runs.

Figure 3. Example run of Algorithm 1.

Task	Template	Verbalizer
SST-2	It was [MASK] . $< S_1 >$	bad / good
	$< S_1 >$. All in all, it was [MASK].	bad / good
	Just [MASK]! $< S_1 >$	bad / good
	In summary, the movie was [MASK].	bad / good
QQP	$< S_1 >$ [MASK], $< S_2 >$	No / Yes
	$< S_1 >$ [MASK], I want to know $< S_2 >$	No / Yes
	$< S_1 >$ [MASK], but $< S_2 >$	No / Yes
	$< S_1 >$ [MASK], please, $< S_2 >$	No / Yes
MNLI	$< S_1 >$? [MASK], $< S_2 >$	Wrong/Right/Maybe
	$< S_1 >$? [MASK], $< S_2 >$"	No/Yes/Maybe
	$< S_1 >$? [MASK], $< S_2 >$	Wrong/Right/Maybe
RTE	"$< S_2 >$? [MASK], $< S_1 >$"	No/Yes
	"$< S_2 >$? [MASK]. $< S_1 >$	No/Yes
	"$< S_1 >$? [MASK]. $< S_2 >$	No/Yes
QNLI	$< S_1 >$? [MASK], $< S_2 >$	No/Yes
	$< S_1 >$? [MASK], $< S_2 >$	Wrong/Right
	"$< S_1 >$? [MASK], $< S_2 >$"	Wrong/Right No/Yes

Table 2. Manual template and verbalizer pairs. $< S_1 >$ and $< S_2 >$ are the input sentences.

4 Experiments

4.1 Experimental setup

Tasks and Datasets.

In the experiments, we utilize 8 tasks from popular benchmarks to thoroughly evaluate the utility, robustness, and efficiency of SecPE. I) Benign NLP tasks. We evaluate SecPE on six tasks from the **GLUE** benchmark. In detail, the evaluated tasks are (1) SST-2; (2) QQP; (3) MNLI-matched; (4) MNLI-mismatched, (5) RTE, and (6) QNLI—range, which range from sentiment analysis to question answering, diversifying in different inference data formats from sentences to pairs of sentences.

II) Adversarial NLP tasks. We evaluate the robustness of SecPE on six adversarial tasks in the Adversarial-GLUE (**AdvGLUE**) benchmark [32], which are adversarial counterparts to the above benign GLUE tasks. The AdvGLUE benchmark is enriched with task-specific adversarial examples generated by 14 different textual attack methods, coming from different adversarial perturbation strategies including word-level, sentence-level, and human-generated. Recognizing the potential problem of invalid adversarial constructs identified by Wang et al. [32], where up to 90% of automatically generated examples may be flawed, we also incorporate human validation. This step allows for a more accurate and robust evaluation of SecPE by ensuring that the adversarial examples in our benchmark are legitimate and that the perturbations maintain the integrity of the original task.

III) Arithmetic reasoning tasks. We evaluate the self-consistency of SecPE on two arithmetic reasoning benchmarks: GSM8K [6] and MultiArith [24]. GSM8K contains grade-school-level mathematical word problems requiring models to perform complex arithmetic reasoning and multi-step calculations. MultiArith contains multiple arithmetic operations within a single problem, testing a model's ability to comprehend and execute a sequence of calculations, reflecting the complexity of mathematical reasoning needed for higher accuracy in various problem-solving contexts.

IV) Zero-shot image classification tasks. We also evaluate SecPE on ImageNet and its variant test sets ImageNet-R , ImageNet-A , ImageNet-Sketch and ImageNet-V2. We also evaluate on CIFAR10 and CIFAR100, which are fine-grained classification datasets.

Private Inference Implementation. We develop encryption functions with C++ and integrate the SEAL library for RNS-CKKS homomorphic encryption. To improve performance on Intel CPUs, we include HEXL acceleration. Our configuration adheres to homomorphic encryption standards, setting the polynomial degree to $N = 2^{16}$ and the ciphertext modulus to 1763 bits for 128-bit security. We set a multiplicative depth of $L = 35$ and a bootstrapping depth of $K = 14$, resulting in an effective multiplicative depth of 21.

4.2 Evaluation Results on GLUE and Adversarial GLUE Tasks

For tasks within the GLUE and AdvGLUE benchmarks, we use the ALBERT-XXLarge-v2 model [15] to generate different contextual representations. This combined text is fed into the model to obtain the language model results. This method allows us to assess the relationship between questions and their corresponding answers, taking

advantage of the model's pre-trained capabilities.

In Table 1, we present evaluation results on GLUE and AdvGLUE tasks, reporting metrics F1 score for QQP and accuracy for the other five tasks). BERT is used as the large pre-trained language model. For baselines LM-BFF and PET, we implement the same private ALBERT-xxlarge-v2 for fair comparison. For the baseline methods, we compare SecPE with LM-BFF [8], and PET [26].

- **LM-BFF [8]:** It involves concatenating the input example, which is modified to follow the prompting template with a `[MASK]` in place of the verbalizer, with semantically similar examples. During inference, LM-BFF ensembles the predictions made by concatenating the input example with all demonstrations from the few-shot training set. (i.e., demonstrations) from the few-shot training set. For each test example, we ensemble the predictions over different possible sets of demonstrations. we perform random sampling and subsequent training of LM-BFF for 5 times and 1000 training steps, for each task.
- **PET [26]:** It is a simple prompt-based few-shot fine-tuning approach where the training examples are converted into templates, and the `[MASK]` tokens are used to predict the verbalizer, which indicates the output label. To understand the role of using multiple prompts in robustness, we use PET to fine-tune models with different template-verbalizer pairs and ensemble their predictions during inference. The pairs used for different tasks are listed in Table 2. We train the model on four different sets of manual template-verbalizer pairs for 250 training steps.

According to Table 1, we have the following experiment results:

- Compared with prompt ensembles without privacy preservation, SecPE exhibits almost no accuracy loss on GELU and AdvGELU benchmarks. This suggests that SecPE is capable of maintaining both high utility and robustness while providing privacy protection.
- Compared with the private inference of a single prompt template, SecPE has demonstrated better adversarial robustness than LM-BFF(Ciphertext).

4.3 Comparison on Arithmetic Reasoning Tasks

For reasoning tasks such as MultiArith and GSM8K, we used the GPT-3 model, specifically the code-davinci-001 variant [4]. This model was chosen for its advanced ability to handle complex language patterns and to generate coherent, contextually relevant text completions.

The Self Consistency approach employs an array of diverse reasoning pathways, each of which may lead to a different final answer. To identify the optimal answer, we marginalize out the sampled reasoning pathways using a voting verifier (aggregate-then-argmax) as described in [17], thereby determining the most consistent answer in the final answer set.

Under the SecPE framework, we have implemented Self Consistency's privacy inference. The baseline to which we are comparing is the chain of reasoning with greedy decoding [35]. The accuracy of the ciphertext inference is similar and much higher than the baseline when compared to the self-consistency inference results under plaintext. Figures 4 and 5 show the performance on GSM8K and MultiArith with different numbers of inference paths.

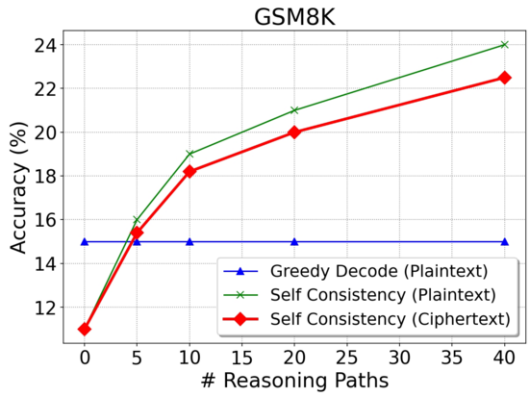

Figure 4. Performance on GSM8K with the different number of reasoning paths.

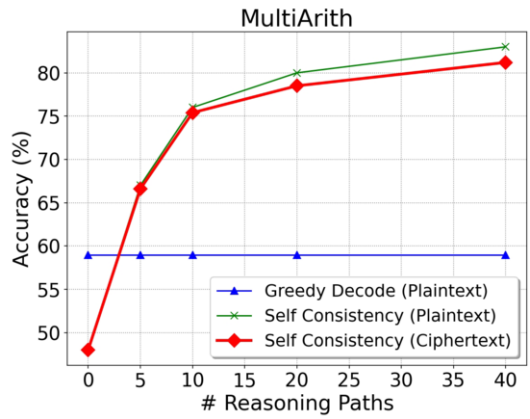

Figure 5. Performance on MultiArith with the different number of reasoning paths.

4.4 Comparison on text-image models

We extend SecPE to Text-Image Models and test the accuracy across 7 tasks from CIFAR, ImageNet, and their variants. We tested zero-shot accuracy with different number of prompts under the CLIP ViT-B/16 model. This model integrates the vision transformer architecture with large language model processing methodologies. This configuration utilizes the "Base" model variant with input patches sized at 16×16, facilitating the processing of visual data through self-attention mechanisms that are typically reserved for textual data.

We tested with different numbers of prompts:

- **1 prompt.** We used CLIP's most commonly used template "A photo of `[MASK]`."
- **80 prompts.** We use the set of 80 prompts designed by [22] for ImageNet.
- **247 prompts.** We constructed the set of 247 prompts using GPT-4, such as "this is a photo of `[MASK]`", "A drawing of a `[MASK]`", etc.

Table 3 shows the zero-shot accuracy on CIFAR-10, CIFAR-100, ImageNet, and its variants. Compared with plaintext inference, the ciphertext inference accuracy of SecPE and Phoenix is slightly lower, which is caused by errors introduced by homomorphic encryption calculations. Experiments show that compared to Phoenix, SecPE's accuracy is higher. For example, in the setting of 247 prompts, SecPE's accuracy is 97.1%, which is only 0.4% lower

Setting	Method	ImageNet	IN-A	IN-R	IN-Sketch	IN-V2	CIFAR-10	CIFAR-100
1 prompt	Plaintext	66.37	47.47	73.78	45.84	60.46	96.2	83.1
	Phoenix (Ciphertext)	66.24	47.38	73.58	45.21	60.44	95.6	82.5
	SecPE (Ciphertext)	66.31	47.43	73.76	45.84	60.44	95.8	82.9
80 prompts	Plaintext	67.63	49.37	77.38	46.95	61.39	96.8	84.3
	Phoenix (Ciphertext)	66.35	47.88	75.75	45.26	60.50	95.8	82.7
	SecPE (Ciphertext)	67.42	49.29	77.11	46.57	61.08	96.4	83.8
247 prompts	Plaintext	68.60	49.63	77.62	47.99	62.21	97.5	87.9
	Phoenix (Ciphertext)	66.71	48.05	75.82	46.07	60.18	95.8	83.1
	SecPE (Ciphertext)	68.33	49.18	77.01	47.38	61.98	97.1	87.6

Table 3. Zero-shot accuracy on CIFAR-10, CIFAR-100, ImageNet and its variants.

than the plaintext inference accuracy. However, Phoenix's accuracy is only 95.8%, 1.7% lower.

It can be demonstrated that each multiplication and rotation operation of RNS-CKKS will introduce a component of the error. The polynomial fitting of the sign operation necessitates a large number of multiplication calculations. Under the setting of 247 prompts, Phoenix requires 248 sign operations and ciphertext rotation, whereas SecPE only requires 8 sign operations and ciphertext rotation, so the error of SecPE will be significantly smaller.

4.5 *Efficiency Comparison*

Figure 6 illustrates the efficiency comparison of SecPE with Phoenix [12] under different input dimensions. In particular, we focus on the essential ARGMAX operation, which incurs one of the major overheads of prompt ensemble under private inference. For an input length of n, Phoenix [12] adopts a sequential comparison approach to obtain the sign bit, resulting in $(n+1)$ $Sign$ operations and $(n+1)$ ciphertext rotations. In contrast, SecPE's Algorithm 1 only requires $(\log n+1)$ $Sign$ operations and $(\log n+1)$ ciphertext rotations. This significantly reduces the execution time, which is depicted in Figure 6. For the input length of 256, SecPE achieves 20.8× speedup for ARGMAX.

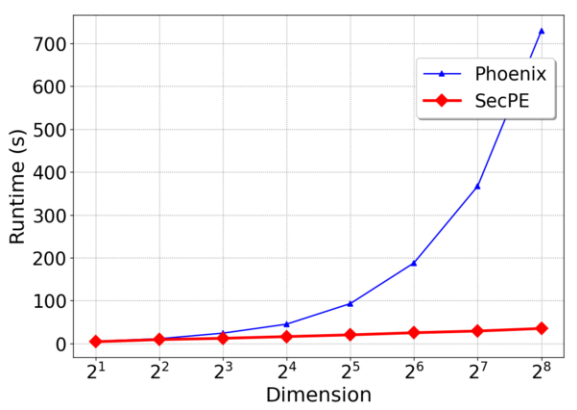

Figure 6. Performance of ARGMAX on RNS-CKKS for different dimensions of input.

Figure 7 shows the time distribution of different building blocks in SecPE. Experiments show that ARGMAX is an bottleneck of the prompt ensembling in ciphertext. The runtime overhead of argmax is even longer than the overhead of privacy LLM inference. SecPE significantly reduces the ARGMAX computation time

Figure 7. Comparison of Total Runtime and Argmax Computation Time between Phoenix and SecPE.

from 2060s to 891s, enhancing the overall efficiency. Therefore, SecPE incurs an additional cost of only 3.8% compared to private inference with LLM without prompt ensembling.

It indicates that while Prompt Ensembling requires multiple inference runs, this overhead is justified. Despite the additional computational cost, as visualized by the substantial slice of the pie chart allocated to LLM inference, the benefits of Prompt Ensembling cannot be overstated. The improved robustness and accuracy provided by multiple inferences, where different prompts are evaluated to derive a final answer, results in more reliable and accurate model performance. This benefit often outweighs the cost of increased inference time, making prompt ensembling a valuable technique in scenarios where high-quality predictions are paramount.

5 Conclusions

We propose SecPE, the first attempt to our knowledge to jointly enable privacy-preserving and adversarial robustness for LLM inference. SecPEsynergizes the strengths of private inference and prompt ensembling, previously studied in isolation, and overcomes the inefficiencies of a naive combination of existing techniques.

Our extensive experiments have shown that SecPE not only maintains high clean accuracy but also significantly improves robustness, all with minimal efficiency overhead compared to existing private inference methods. Thus, SecPE manifests a satisfactory "accuracy-robustness-efficiency" tradeoff. The future work is to take advantage of existing hardware acceleration technology, such as GPU [34] and FPGA [31], which is expected to increase the efficiency by hundreds of times to achieve practical private inference.

Acknowledgements

This work was supported by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (No. 2022C01126).

References

[1] J. U. Allingham, J. Ren, M. W. Dusenberry, X. Gu, Y. Cui, D. Tran, J. Z. Liu, and B. Lakshminarayanan. A simple zero-shot prompt weighting technique to improve prompt ensembling in text-image models. In *International Conference on Machine Learning*, pages 547–568. PMLR, 2023.

[2] J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 587–617. Springer, 2021.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[5] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. A full rns variant of approximate homomorphic encryption. In *Selected Areas in Cryptography–SAC 2018: 25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers 25*, pages 347–368. Springer, 2019.

[6] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[7] N. Dvornik, C. Schmid, and J. Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3723–3731, 2019.

[8] T. Gao, A. Fisch, and D. Chen. Making pre-trained language models better few-shot learners. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.295. URL https://aclanthology.org/2021.acl-long.295.

[9] C. Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[10] K. Han and D. Ki. Better bootstrapping for approximate homomorphic encryption. In *Cryptographers' Track at the RSA Conference*, pages 364–390. Springer, 2020.

[11] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, and T. Zhang. Iron: Private inference on transformers. *Advances in Neural Information Processing Systems*, 35:15718–15731, 2022.

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

[13] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1651–1669, 2018.

[14] J. D. M.-W. C. Kenton and L. K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2, 2019.

[15] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[16] E. Lee, J.-W. Lee, J. Lee, Y.-S. Kim, Y. Kim, J.-S. No, and W. Choi. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In *International Conference on Machine Learning*, pages 12403–12422. PMLR, 2022.

[17] Y. Li, Z. Lin, S. Zhang, Q. Fu, B. Chen, J.-G. Lou, and W. Chen. Making language models better reasoners with step-aware verifier. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.291. URL https://aclanthology.org/2023.acl-long.291.

[18] Y. Li, Y.-L. Tsai, C.-M. Yu, P.-Y. Chen, and X. Ren. Exploring the benefits of visual prompting in differential privacy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5158–5167, 2023.

[19] C. Mauran. Whoops, samsung workers accidentally leaked trade secrets via chatgpt. *Mashable [online]. Dostupné z: https://mashable.com/article/samsungchatgpt-leak-details*, 2023.

[20] Natasha Lomas. Italy orders chatgpt blocked citing data protection concerns. https://techcrunch.com/2023/03/31/chatgpt-blocked-italy/, 2023. Accessed: 2023-05-28.

[21] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider. Bolt: Privacy-preserving, accurate and efficient inference for transformers. *IEEE Symposium on Security and Privacy (SP)*, 2024.

[22] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[23] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[24] S. Roy and D. Roth. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*, 2016.

[25] T. Schick and H. Schütze. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*, 2020.

[26] T. Schick and H. Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In P. Merlo, J. Tiedemann, and R. Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online, Apr. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.20. URL https://aclanthology.org/2021.eacl-main.20.

[27] T. Schick, H. Schmid, and H. Schütze. Automatically identifying words that can serve as labels for few-shot text classification. *arXiv preprint arXiv:2010.13641*, 2020.

[28] X. Shen, B. Tan, and C. Zhai. Privacy protection in personalized search. In *ACM SIGIR Forum*, volume 41, pages 4–17. ACM New York, NY, USA, 2007.

[29] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[30] W. Z. Srinivasan, P. Akshayaram, and P. R. Ada. Delphi: A cryptographic inference service for neural networks. In *Proc. 29th USENIX Secur. Symp*, pages 2505–2522, 2019.

[31] A. Viand, P. Jattke, M. Haller, and A. Hithnawi. {HECO}: Fully homomorphic encryption compiler. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4715–4732, 2023.

[32] B. Wang, C. Xu, S. Wang, Z. Gan, Y. Cheng, J. Gao, A. H. Awadallah, and B. Li. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840*, 2021.

[33] Y. Wang, Y. Chen, Z. Li, Z. Tang, R. Guo, X. Wang, Q. Wang, A. C. Zhou, and X. Chu. Towards efficient and reliable llm serving: A real-world workload study. *arXiv preprint arXiv:2401.17644*, 2024.

[34] Z. Wang, P. Li, R. Hou, Z. Li, J. Cao, X. Wang, and D. Meng. Hebooster: An efficient polynomial arithmetic acceleration on gpus for fully homomorphic encryption. *IEEE Transactions on Parallel and Distributed Systems*, 34(4):1067–1081, 2023.

[35] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[36] H. Xu, B. Liu, L. Shu, and P. S. Yu. Bert post-training for review reading comprehension and aspect-based sentiment analysis. *arXiv preprint arXiv:1904.02232*, 2019.

[37] H. Yan, S. Li, Y. Wang, Y. Zhang, K. Sharif, H. Hu, and Y. Li. Membership inference attacks against deep learning models via logits distribution. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[38] W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, and J. Lin. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*, 2019.

[39] W. Yang, H. Zhang, and J. Lin. Simple applications of bert for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972*, 2019.

[40] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri. Enhanced membership inference attacks against machine learning models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 3093–3106, 2022.

[41] W. Zeng, M. Li, W. Xiong, T. Tong, W.-j. Lu, J. Tan, R. Wang, and R. Huang. Mpcvit: Searching for accurate and efficient mpc-friendly vision transformer with heterogeneous attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5052–5063, 2023.