Security and Safety, Vol. 4, 2025014 (2025)
https://doi.org/10.1051/sands/2025014
sands.edpsciences.org

Research Article OPEN ACCESS

Other Fields

PUMA: Secure inference of LLaMA-7B in five minutes

Ye Dong[1,2], Wen-Jie Lu[1], Yancheng Zheng[1], Haoqi Wu[1], Derun Zhao[1], Jin Tan[1], Zhicong Huang[1], Cheng Hong[1,*], Tao Wei[1], Wen-Guang Chen[1], and Jianying Zhou[3]

[1] *Ant Group, Beijing 100081, China*
[2] *National University of Singapore, Singapore 119260, Singapore*
[3] *Singapore University of Technology and Design, Singapore 487372, Singapore*

Received: 15 July 2025 / Revised: 14 October 2025 / Accepted: 16 October 2025 / Published online: 23 October 2025

Abstract Transformer models (e.g., Bert and GPT) have shown their dominance in machine learning tasks. Many cloud companies have begun to provide services based on Transformer models, examples include translation and text-speech conversion. However, such a service inevitably requires access to the client's data, which might contain sensitive information. Theoretically, running the services under secure multi-party computation (MPC) could protect clients' privacy. However, current MPC frameworks are still limited in terms of model performance, efficiency, deployment, and functionality, especially when facing complex Transformer models. To this end, we propose an MPC framework PUMA to enable secure and efficient Transformer model inference. We first design high-quality approximations for the bottleneck functions in Transformers such as GELU and Softmax, reducing about 20−76% computation and communication costs than state-of-the-art works without performance drop. Then, we provide concrete instantiations for secure Embedding and LayerNorm. These implementations produce correct results and integrate compatible system architectures of cleartext Transformer models. Finally, we conducted extensive experiments on six popular benchmarks: text classification/generation/summarization/translation, audio-to-text, and image-to-text. Results show that PUMA can finish most tasks in several minutes, with comparable model performance (e.g., accuracy) as cleartext, and even evaluate LLaMA-7B in less than 5 minutes to generate 1 token.

Keywords Privacy, Security, Secure Three-Party Computation, Privacy-Preserving Machine Learning, Large Language Models

Citation Dong Y, Lu WJ, Zheng Y, Wu H, Zhao D, Tan J, Huang Z, Hong C, Wei T, Chen WG and Zhou J. PUMA: Secure inference of LLaMA-7B in five minutes. Security and Safety 2025; **4**: 2025014. https://doi.org/10.1051/sands/2025014

1 Introduction

Transformer models [1] have attracted much attention for their high performance in practical tasks [2, 3] and have been widely used in online applications [4–6] under Deep Learning as a Service (DLaaS) paradigm [7]. However, these applications can raise privacy concerns, because they require users to reveal their input data to the service provider. These data may contain sensitive information such as chat logs, personal pictures, or bank transactions.

One solution to address the privacy concerns of Transformer models service is Secure Multi-Party Computation (MPC) [8–10], which can keep data and model weights private during inference. Table 1

* Corresponding author (email: vince.hc@antgroup.com)

Table 1. Representative secure Transformer models inference frameworks. MM is for matrix multiplication, L.N. indicates LayerNorm, and E.M. stands for Embedding. • indicates the used techniques and ○ is for not used. ✔ indicates supporting a feature and ✘ is for not supported ones. M-Mod is for multi-modality. Compared to two-party computation (2PC) frameworks, three-party computation (3PC) ones are usually faster at the cost of additional assumptions of non-colluding servers

	Framework	Techniques				Transformer Functions					Re-Train	M-Mod.	Code
		HE	SS	OT	FSS	MM	GELU	Softmax	L.N.	E.M.			
2PC	Iron [14]	•	•	•	○	✔	✔	✔	✔	✘	Not Req.	✘	✔
	BumbleBee [12]	•	•	•	○	✔	✔	✔	✔	✔	Not Req.	✘	✔
	CipherGPT [15]	•	•	•	○	✔	✔	✔	✔	✔	Not Req.	✘	✘
	BOLT [16]	•	•	•	○	✔	✔	✔	✔	✔	Req.	✘	✔
	NEXUS [23]	•	○	○	○	✔	✔	✔	✔	✘	Not Req.	✘	✔
3PC	SIGMA [13]†	○	○	○	•	✔	✔	✔	✔	✘	Not Req.	✘	✔
	Privformer [17]	○	•	○	○	✔	✘	✘	✔	✔	Req.	✘	✘
	Ditto [20]	○	•	○	○	✔	✔	✔	✔	✔	Req.	✘	✔
	Crypten [24]†	○	•	○	○	✔	✘	✔	✘	✘	Req.	✘	✔
	MPCFormer [11]†	○	•	○	○	✔	✘	✘	✘	✘	Req.	✘	✔
	Puma (Ours)	○	•	○	○	✔	✔	✔	✔	✔	Not Req.	✔	✔

† They work in dealer-based 3PC, where the dealer generates the correlated randomness for two computing parties.

shows many existing works have proposed various ways to support secure Transformer model inference using MPC, but these approaches are still struggling with one or several of the following challenges:

Challenge-❶: High Inference Cost. Most importantly, there are still performance gaps between current MPC-based Transformer inference frameworks and large models used in real world. Due to the high computational and communication overhead, these systems have been limited to relatively small and simple transformer models (*e.g.*, Bert and GPT2 with a few millions of parameters) [11–17]. Recently, MPC-based frameworks [12, 13] have made considerable improvements, and demonstrated the ability to perform secure inference at the scale of LLaMA-7B. However, there remains considerable overhead: For instance, using BumbleBee [12], the server and the client might need more than 13 minutes to generate 1 token for 8 token prompts using LLaMA-7B. SIGMA [13] is fast in the online phase, but requires expensive preprocessing to generate hundreds of gigabytes of function secret sharing (FSS) keys for each query.

Challenge-❷: Retraining (Fine-tuning) Required. To reduce the cost of non-linear functions, several works [11, 17, 18] suggested approximating GELU, Exp, and Softmax using simpler functions like ReLU and quadratic polynomials. These functions are up to an order of magnitude cheaper in MPC but would introduce utility loss to the Transformer model. As a result, they require an extra step of model retraining/fine-tuning. BOLT [16] also applies quantization-aware fine-tuning [19] to counteract the accuracy loss from small bit-width and scales. Ditto [20] utilizes distillation-based fine-tuning to quantize the Transformer models with small bit-widths for better efficiency. However, retraining/fine-tuning is unfriendly for data-limited participants, and might not achieve satisfactory model performance [21].

Challenge-❸: Incompatible Architectures. Works [11, 22] proposed to modify the architecture of Transformer models to accelerate secure inference, *e.g.*, decompose the embedding procedure, reorganize the linear layers, and replace LayerNorm with BatchNorm. Some other works define the Transformer models from scratch in their customized implementation (*e.g.*, C++) [13–17, 23]. They require considerable manual effort to modify and convert an existing cleartext model into an MPC one. Since the machine learning area is fast evolving, and new models are coming out every week, such requirements might not be satisfactory for agile deployment. It will be much more competitive if the service provider could seamlessly load pre-trained cleartext models.

Challenge-❹: Limited Modalities. Existing works [11–18, 22] primarily focus on simple Transformer models (*e.g.*, Bert) and tasks (*e.g.*, classification). However, practical applications often require handling more complex tasks (*e.g.*, text summarization and translation) and diverse modalities (*e.g.*, audio/image-to-text), where the effectiveness of existing works remains unknown.

To summarize, in the field of MPC Transformer-based serving framework, achieving model performance, efficiency, compatible architectures, and diverse modalities are challenging, and people may ask:

Could pre-trained large Transformer models be securely and efficiently evaluated on real-world tasks with comparable performance as cleartext and compatible architectures with existing Transformer libraries, and without further retraining/fine-tuning?

To address these challenges, we propose our PUMA, a fast and accurate end-to-end MPC-secure TransforMer-based serving framework that can support LLaMA-7B. Our contributions are as follows:

- **Protocols for Non-linear Functions of Transformer.** We propose accurate and fast MPC protocols for expensive non-linear functions of Transformer models, including GELU, Softmax, LayerNorm, and Embedding. By designing MPC-friendly yet accurate approximations based on the specialized properties of these functions, we achieve both high accuracy and efficiency. Additionally, our protocols align with the workflows required by standard cleartext Transformer libraries. Some works such as BumbleBee [12] and Ditto [20] also follow this way, but PUMA is the first to lead this trend.
- **End-to-End Framework Compatible with Cleartext.** Benefiting from the high-accuracy and aligned-workflows of our protocols, we implement all the layers required by the Transformer in MPC, following the workflows of cleartext Transfomer libraries. This allows us to easily load and securely evaluate the pre-trained cleartext Transformer models (*e.g.*, downloaded from Hugging Face). To our best knowledge, PUMA is the first open-sourced MPC solution that supports accurate inference of pre-trained Transformer models without further modifications such as retraining/fine-tuning. We opensource PUMA at https://github.com/AntCPLab/puma_benchmarks.
- **Fast and Accurate Secure Transformer Model Inference.** We make extensive experiments on 10 Transformer models and 8 datasets with devise modalities, including text classification, generation, summarization, translation, audio-to-text, and image-to-text. Results show PUMA's model performance is similar to cleartext ones' and is about 2× faster than CRYPTEN (note CRYPTEN does not achieve comparable model performance as PUMA) and SecretFlow-SPU. PUMA can even evaluate LLaMA-7B in less than 5 minutes to generate one word given 8 token prompts.

Remark. We are the first to propose accurate and efficient approximations for complex activation functions in Transformer models. Following our work, several recent studies [12, 16, 20, 23] have proposed their own approximations or optimized our methods in their respective settings, and have cited our preprint version [25].

Organization. We introduce the background in Section 2. The overview of our architecture is given in Section 3 and the detailed design is in Section 4. We analyze the experiments in Section 6. Related works are summarized in Section 7 and our conclusion is in Section 8.

2 Preliminary and background

2.1 Notations

The main used notations are as follows: P_i represents the i-th computing party, $i \in \{0, 1, 2\}$. The uppercase bold letter $\mathbf{X}$ is used for matrices, and the lowercase bold letter $\mathbf{x}$ denotes vectors. $\mathbf{x}[j]$ denotes the j-th element of vector $\mathbf{x}$, while lowercase letter x is used for scalar values. $\mathbb{Z}_{2^\ell}$ denotes the discrete ring modulo 2^ℓ, $\mathbb{R}$ denotes real numbers. $[\![\cdot]\!]$ is used for 2-out-of-3 replicated secret sharing [26, 27].

2.2 Transformer model

Transformer models have achieved remarkable success in language understanding [2, 28–30], vision understanding [3, 25, 31], and etc. Two popular variants are Bert (Bidirectional Encoder Representations from Transformers) [28] and GPT (Generative Pre-Trained models) [2]. A Transformer model [1] mainly consists of Embedding, Attention, Feed-Forward Network, and LayerNorm sub-layers. Given a token (*e.g.*, a word) id, the Embedding layer maps it to a hidden vector representation by querying the Embedding table. Below, we have given the details of the other sub-layers:

Attention. Given inputs $(\mathbf{Q}, \mathbf{K}, \mathbf{V})$, the Attention function is computed as $\mathsf{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathsf{Softmax}(\mathbf{Q} \cdot \mathbf{K}^{\mathsf{T}} + \mathbf{M}) \cdot \mathbf{V}$, where $\mathbf{M}$ can be viewed as a bias matrix. Besides, Multi-Head Attention [1] can attend to information from different representation subspaces at different positions in parallel.

Feed-Forward Network (FFN). FFN is applied to each position separately and identically. This consists of two linear transformations with an activation in between. Given input $\mathbf{x}$ and parameters $\{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2\}$, FFN can be formalized $\mathsf{FFN}(\mathbf{x}) = \mathbf{W}_2\mathsf{Act}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$, where Act denotes the activation function, and the common activation functions in Transformer include GELU and SiLU [32], which are much more complex than ReLU. Note that the parameters of linear transformations are different from layer to layer.

LayerNorm. Given vector $\mathbf{x} \in \mathbb{R}^n$, LayerNorm is defined as: $\mathsf{LayerNorm}(\mathbf{x})[j] = \gamma \cdot \frac{\mathbf{x}[j]-\mu}{\sqrt{\sigma}} + \beta$, where (γ, β) are trained parameters, mean $\mu = \frac{\sum_{j=1}^{n} \mathbf{x}[j]}{n}$, and standard deviation $\sigma = \sum_{j=1}^{n}(\mathbf{x}[j]-\mu)^2$. Compared to BatchNorm, LayerNorm is significantly more complex and expensive during secure inference. This is because the mean and standard deviation (μ, σ) depend on the test data and cannot be accurately approximated by the training dataset, unlike BatchNorm [33].

2.3 2-out-of-3 replicated secret sharing

Secret sharing is widely used in MPC-based machine learning, where privacy is ensured by keeping all the intermediate values shared between multiple non-colluding computing parties. Here we choose 2-out-of-3 replicated secret sharing (RSS) [27] for its high efficiency. In RSS, a secret value $x \in \mathbb{Z}_{2^\ell}$ is shared by three random values $x_0, x_1, x_2 \in \mathbb{Z}_{2^\ell}$ with $x = x_0 + x_1 + x_2 \pmod{2^\ell}$, and party P_i gets (x_i, x_{i+1}) (denoted as $[\![x]\!]_i$). Without special declaration, we compute in $\mathbb{Z}_{2^\ell}$ and omit $\pmod{2^\ell}$ for brevity. In the case of $\ell > 1$ (*e.g.*, $\ell = 64$) which support arithmetic operations (*e.g.*, $+$, $-$, and $\cdot$), we refer to this type as *Arithmetic Sharing* and use notation $[\![\cdot]\!]$. *Boolean Sharing* ($[\![\cdot]\!]^{\mathsf{B}}$) refers to $\ell = 1$ where $(+, -)$ and $\cdot$ are respectively replaced by bit-wise $\oplus$ and $\wedge$.

2.3.1 Addition and multiplication

Let (c_1, c_2, c_3) be public constants, and $([\![x]\!], [\![y]\!])$ be two secret-shared values. The secure addition and multiplication procedures are as follows:

Addition. $[\![c_1x + c_2y + c_3]\!]$ can be computed as $(c_1x_0 + c_2y_0 + c_3, c_1x_1 + c_2y_1, c_1x_2 + c_2y_2)$, where P_i can compute its share locally. When $(c_1 = 1, c_2 = 1, c_3 = 0)$, we get $[\![x + y]\!]$.

Multiplication. In secure multiplication $\mathcal{F}_{\mathrm{Mul}}([\![x]\!], [\![y]\!])$, parties follow steps: i) First, P_i computes $z_i = x_iy_i + x_{i+1}y_i + x_iy_{i+1}$ locally, ii) Parties then perform *re-sharing* by letting P_i sends $z'_i = \alpha_i + z_i$ to P_{i-1}, where $\alpha_0 + \alpha_1 + \alpha_2 = 0$ (P_i can generate α_i using pseudorandom generators with negligible overhead as [27]). iii) Finally, $\{(z'_0, z'_1), (z'_1, z'_2), (z'_2, z'_0)\}$ form the 2-out-of-3 replicated secret shares of $[\![x \cdot y]\!]$.

2.3.2 3-Party functionalities

In addition to addition and multiplication, PUMA invokes several other 3-party functionalities from the state-of-the-art works [27, 34, 35] as follows:

- $[\![z]\!] = \mathcal{F}_{\mathsf{Mul_{BA}}}([\![b]\!]^{\mathsf{B}}, [\![x]\!])$, s.t. $z = b \cdot x$
- $[\![z]\!] = \mathcal{F}_{\mathsf{Square}}([\![x]\!])$, s.t. $z = x^2$
- $[\![z]\!]^{\mathsf{B}} = \mathcal{F}_{\mathsf{Eq}}([\![x]\!], [\![y]\!])$, s.t. $z = 1\{x = y\}$
- $[\![z]\!]^{\mathsf{B}} = \mathcal{F}_{\mathsf{LT}}([\![x]\!], [\![y]\!])$, s.t. $z = 1\{x < y\}$
- $[\![z]\!] = \mathcal{F}_{\mathsf{Recip}}([\![x]\!])$, s.t. $z = 1/x$
- $[\![z]\!] = \mathcal{F}_{\mathsf{rSqrt}}([\![x]\!])$, s.t. $z = 1/\sqrt{x}$
- $[\![z]\!] = \mathcal{F}_{\mathsf{Max}}([\![\mathbf{x}]\!])$, s.t. $z = \mathsf{maximum}(\mathbf{x})$
- $[\![z]\!] = \mathcal{F}^f_{\mathsf{Trunc}}([\![x]\!])$, $z = \lfloor x/2^f \rfloor + E$, $E \in \{0, \pm 1\}$

To simplify the protocol description and security proofs, we describe PUMA using the hybrid model [36].

2.3.3 Fixed-point representation and truncation

Real numbers should be encoded into fixed-point numbers before being represented in finite rings/fields. To avoid overflow, $\mathcal{F}^f_{\mathsf{Trunc}}$ has to be used after each fixed-point multiplication to truncate the least f

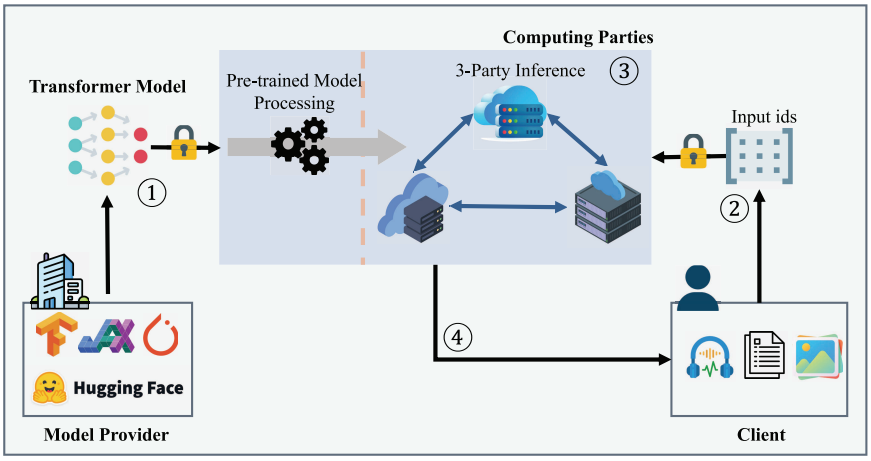

Figure 1. The overview of Puma's workflow. ① Model provider secret-shares the model among the computing parties. ② Client generates the input-ids for the input file, secret-share the input-ids, and sends the shares to computing parties. ③ The servers process and evaluate the model on input-ids securely using 3-party protocols. ④ The servers return results' shares to the client, who performs reconstruction

bits securely [37]. For simpler description, we include $\mathcal{F}^{f}_{\mathsf{Trunc}}$ in $\mathcal{F}_{\mathsf{Mul}}$ and $\mathcal{F}_{\mathsf{Square}}$ by default and do not explicitly mention it in our protocol designs.

The above operations can be easily extended to vectors and matrices, and we use the same notation for vector and matrix operations for simplicity [27, 38].

3 Overview of Puma's architecture

To achieve secure inference of Transformer models, Puma defines three kinds of roles: model owner, client, and three non-colluding computing parties. The computing parties should be hosted by different cloud vendors, *e.g.*, Amazon AWS, Google Cloud, and Microsoft Azure. As illustrated in Figure 1, the model provider and client send their respective models and inputs to the computing parties (*i.e.*, P_0, P_1, and P_2) in a secret-shared form (steps ① and ②). Then, the computing parties process and load the model into the MPC world seamlessly, execute MPC protocols on top of the secret-shared data (③), and send the shared output back to the client, who reconstructs the shares and gets the final result (④). Note that an entity could play multiple roles, *e.g.*, the model could be provided by one of the cloud vendors, here we describe them separately for generality. Since all the data are kept secret-shared during the whole process, and it is reasonable to assume that big vendors will not collude with each other to reconstruct client's information, data privacy could be guaranteed.

For the cost evaluation of Puma, we have the following declarations:

- For step ①, the model provider can secret-share the model before service, and the same model can be used to provide services many times. Therefore, the amortized costs of model sharing can be negligible;
- For steps ② and ④, we only need little costs between the computing servers and client to communicate secret shares of inputs and results, which are also negligible compared to the costs of 3-party inference.
- With the above analysis in mind, we mainly focus on the communication and running time of the 3-party inference (step ③).

During the secure 3-party inference process, a key invariant is maintained: For any layer, the computing parties always start with 2-out-of-3 replicated secret shares of the previous layer's output and the model parameters[1], and end with 2-out-of-3 replicated secret shares of this layer's output. As the shares do not leak any information to each party, this ensures the layers can be sequentially combined for arbitrary depths to obtain a secure computation scheme for any Transformer models.

[1] In Puma, we focus on the privacy of clients' input and model weights. We do not protect the privacy of the hyper-parameters, *e.g.*, model architecture.

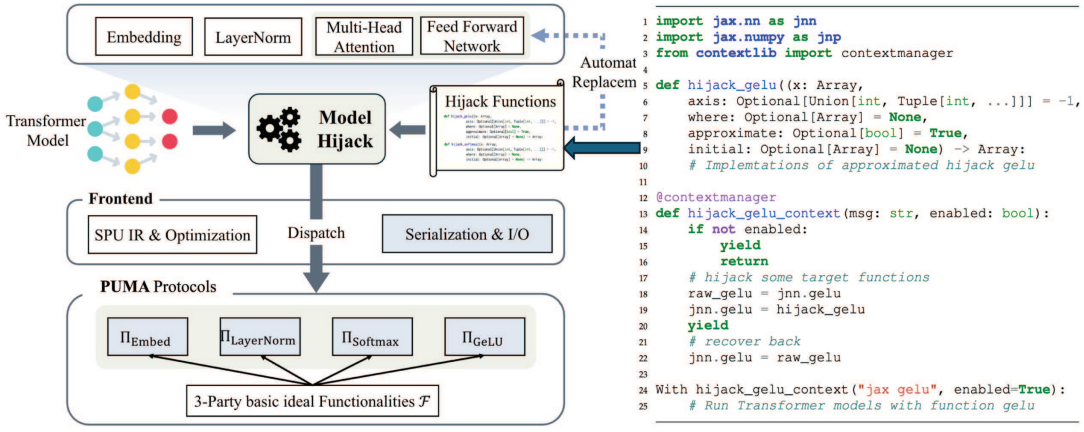

Figure 2. Pre-trained Transformer model processing, loading, and functions hijack in PUMA

Threat Model. Following works [11, 27], PUMA is secure against a semi-honest (a.k.a., honest-but-curious) adversary that corrupts no more than one of three computing parties, such an adversary will follow the protocol specifications, but may try to learn other's private information during the protocol. Note that PUMA cannot defend against attacks based on inference results, and the mitigation methods (*e.g.*, differential privacy [39]) fall outside the scope of this study.

Definition 1. [Semi-Honest Security] Let Π be a three-party protocol running in real-world and $\mathcal{F} : (\{0,1\}^n)^3 \to (\{0,1\}^m)^3$ be the ideal randomized functionality. We say Π securely computes $\mathcal{F}$ in presence of a single semi-honest adversary if for every corrupted party P_i $(i \in \{0,1,2\})$ and every input $\mathbf{x} \in (\{0,1\}^n)^3$, there exists an efficient simulator $\mathcal{S}$ such that:

$$\{\mathsf{view}_{i,\Pi}(\mathbf{x}), \mathsf{output}_{\Pi}(\mathbf{x})\} \overset{c}{\approx} \{\mathcal{S}(I, \mathbf{x}_i, \mathcal{F}_i(\mathbf{x})), \mathcal{F}(\mathbf{x})\},$$

where $\mathsf{view}_{i,\Pi}(\mathbf{x})$ is the view of P_i in the execution of Π on $\mathbf{x}$, $\mathsf{output}_{\Pi}(\mathbf{x})$ is the output of all parties, and $\mathcal{F}_i(\mathbf{x})$ denotes the i-th output of $\mathcal{F}(\mathbf{x})$.

4 Design of PUMA

In this section, we first present our consideration about loading pre-trained Transformer models seamlessly, and then give optimizations and designs for the functions of Transformer models. Note that linear layers such as matrix multiplication are straightforward in 3-party computation, so we omit them for brevity.

4.1 Considerations of seamless model loading

As shown in the step ① of Figure 1, the parties must load and convert the pre-trained Transformer models from cleartext into MPC world for subsequent service. While some works tried to offer the capability to load pre-trained models, they require considerable efforts to re-define the model from scratch [13–16] or modify the models' architecture [11, 24]. *Note that these efforts are required for each model.* These present significant challenges in practical deployments, especially if there are frequent new models. In contrast, once set up, PUMA could load new models without any tailored modifications.

As illustrated in Figure 2, we present how to process and load a pre-trained Transformer model into PUMA, which is compatible with different mainstream cleartext ML frameworks. Consequently, given a Transformer program defined in JAX, PUMA will implicitly call JAX API and SecretFlow-SPU [40] frontend (*i.e.*, IR and Optimization) to convert the functions of Transformer models into the backend MPC protocols, *e.g.*, the function GELU can be dispatched as addition, multiplication, and tanh. However, to execute the Transformer models defined in public libraries (*e.g.*, Hugging Face transformers [41]) efficiently, we integrate the following improvements and optimizations.

4.1.1 Serialization and I/O

Existing serialization libraries of SecretFlow-SPU, such as Protobuf [42] and FlatBuffers [43], only support data trunks with size up to 2 GB, which can only support small size Transformer models (*e.g.*, Bert-Base) and is not sufficient for large MPC-based Transformer models (*e.g.*, LLaMA-7B). To address this problem, we propose an optimization in terms of the serialization and I/O sub-systems. Concretely, the system could automatically divide and serialize overly large secret-shared structures into smaller chunks when communicating or performing I/O operations.

4.1.2 Hijack complex function

SecretFlow-SPU currently provides support for vanilla JAX models implemented entirely in Python. It achieves this by directly dispatching model functions as a series of fundamental backend protocols. *e.g.*, GeLU is dispatched as defined in jax.nn.gelu. When evaluating Transformer models securely in SecretFlow-SPU, an evident limitation arises from its lack of consideration for the MPC-friendly approximations for complex functions (*e.g.*, GELU and Softmax). Consequently, developers proposing new approximation methods have to implement a new function from scratch, then re-direct all the calls to their new function, which is a task that poses significant complexity, particularly for those developers without enough MPC expertise. To address this limitation, we exploit decorator contextmanager to hijack the original functions. This allows for the seamless integration of our proposed approximated procedures into the end-to-end pipeline. The process unfolds as follows:

- **Automatic Replacement.** PUMA automatically substitutes the computation of hijacked functions with our proposed approximated procedures. These procedures are meticulously designed to leverage basic operations while ensuring MPC-friendliness and accuracy.
- **Dispatch to MPC Primitives.** Following the automatic replacement, PUMA proceeds to dispatch the computation of our proposed approximated procedures into the corresponding MPC protocols. This dispatch is crucial for ensuring compatibility with the underlying secure computation framework.

We show how to hijack GELU functions in Figure 2, and our approximated methods are in Section 4.2.

4.1.3 Compatible function support

As outlined in Section 1, just running Transformer models under MPC is insufficient for an agile serving framework. It is essential to ensure the model architectures in MPC are compatible with the cleartext ones. Current works fail to do this, *e.g.*, [11, 24] necessitate the client to generate a one-hot vector using the token id locally. This deviates from a cleartext Transformer workflow where the one-hot vector is generated inside the model. As a result, they have to carefully strip off the one-hot step from the pre-trained models, and add the step to the client side, which is incompatible with the cleartext system and could be an obstacle for deployment. To circumvent these problems, we propose faithful implementations of secure Embedding and LayerNorm for Transformer models inference, which are given in Section 4.3.

4.2 Optimizations of secure GELU and Softmax

GELU and Softmax are much more expensive than the other functions in secure Transformer inference [11, 14, 17]. To reduce their costs while preserving model performance (*e.g.*, do not need fine-tuning/re-training), we propose several approximation and optimizations for GELU and Softmax.

4.2.1 Optimizations of secure GELU

We give our design of approximated piece-wise polynomials for GELU function, and then propose their secure evaluation protocol.

Approximated Polynomials. Most of the current approaches view the GELU function as a composition of smaller functions and try to optimize each piece of them, making them miss the chance of optimizing the secure GELU as a whole. Given the GELU function:

$$\begin{aligned}\mathsf{GELU}(x) &= \frac{x}{2}\cdot\left(1+\tanh\left(\sqrt{\frac{2}{\pi}}\cdot\left(x+0.044715\cdot x^3\right)\right)\right),\\ &\approx x\cdot\mathsf{sigmoid}(0.071355\cdot x^3+1.595769\cdot x)\end{aligned} \tag{1}$$

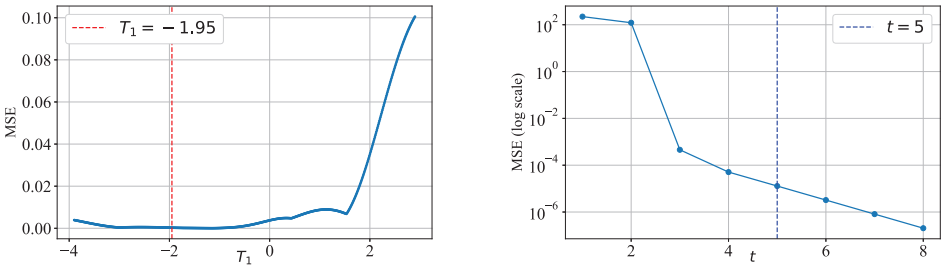

Figure 3. The Mean Square Error (MSE) for different hyperparameters of approximated GELU and negative exponentiation. 3(a) is for MSE with different T_1 in domain $[-4, 3]$, and 3(b) is for MSE with t for domain $[-14, 0]$

these approaches [14, 44] focus either on designing approximate protocols for function tanh or using existing general MPC protocols of exponentiation and reciprocal for sigmoid.

However, we observe the fact that the GELU function is almost linear on the two sides (*i.e.*, $\mathsf{GELU}(x) \approx 0$ for $x < -4$ and $\mathsf{GELU}(x) \approx x$ for $x > 3$). Within the short interval $[-4, 3]$ of GELU, we suggest a piece-wise approximation of low-degree polynomials are more efficient and easy-to-implement choice for its secure protocol. Concretely, our piece-wise low-degree polynomials for GELU are shown in equation (2):

$$\mathsf{AppGELU}(x) = \begin{cases} 0, & x < T_0 \\ F_0(x), & T_0 \le x < T_1 \\ F_1(x), & T_1 \le x \le T_2 \\ x, & x > T_2 \end{cases}, \tag{2}$$

where the segments parameters $(T_0 = -4, T_1 = -1.95, T_2 = 3)$. For threshold T_1 and the polynomials F_0 and F_1 as follows: We first attempted to fit the interval $[-4, 3)$ using a single low-degree polynomial (degree ≤ 6), but the approximation error remained high. We then adopted a two-polynomial strategy, employing a binary search with a stride 0.01 to identify T_1. Balancing accuracy and polynomial degree, we selected $T_1 = -1.95$ as shown in Figure 3(a), and we can get sparse polynomials by using library numpy.ploufit[2] as in equation (3). The above simple polyfit works very well and our mean error is less than 0.003 for $\mathbb{Z}_{2^{64}}$ with $f = 18$ fractional bits. We compare the absolute errors of our and prior approximation methods for GELU functions in Figure 4.

$$\begin{cases} F_0(x) = -0.011034134030615728x^3 - 0.11807612951181953x^2 \\ \qquad\quad -0.42226581151983866x - 0.5054031199708174 \\ F_1(x) = 0.0018067462606141187x^6 - 0.037688200365904236x^4 \\ \qquad\quad +0.3603292692789629x^2 + 0.5x + 0.008526321541038084 \end{cases}, \tag{3}$$

We focus on specialized approximations for nonlinear functions in Transformer models. For low-degree polynomial approximations towards general functions, please refer to [45].

Secure AppGELU Evaluation. Given AppGELU as equation (3), we need secure segment selection and evaluation of F_0 and F_1 to reach the final protocol. The secure segment selection can be achieved by Less-Than functionality $\mathcal{F}_{\mathsf{LT}}$. For secure evaluation of F_0 and F_1, we exploit $\mathcal{F}_{\mathsf{Square}}$ to compute all even-power terms $\{[\![x^2]\!], [\![x^4]\!], [\![x^6]\!]\}$. Also, we require $\mathcal{F}_{\mathsf{Mul}}$ to compute $[\![x^3]\!]$. Finally, we use the $\mathcal{F}_{\mathsf{Mul_{BA}}}$ to select the result securely. Formally, our secure GELU protocol Π_{GELU} is constructed in Algorithm 1.

4.2.2 Optimizations of secure Softmax

In the function $\mathsf{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathsf{Softmax}(\mathbf{Q} \cdot \mathbf{K}^{\mathsf{T}} + \mathbf{M}) \cdot \mathbf{V}$, the key challenge is computing function Softmax. For the sake of numerical stability, the Softmax function is computed as

$$\mathsf{Softmax}(\mathbf{x})[j] = \frac{\exp(\mathbf{x}[j] - \bar{x} - \epsilon)}{\sum_j \exp(\mathbf{x}[j] - \bar{x} - \epsilon)}, \tag{4}$$

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

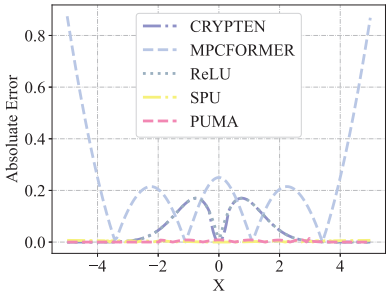

Figure 4. The absolute errors of our and prior methods for GELU functions

Algorithm 1 Protocol Π_{GELU}

Input: P_i holds the 2-out-of-3 replicate secret share $[\![x]\!]_i$ for $i \in \{0,1,2\}$
Output: P_i gets the 2-out-of-3 replicate secret share $[\![y]\!]_i$ for $i \in \{0,1,2\}$, where $y = \mathsf{GELU}(x)$.

1: P_0, P_1, and P_2 jointly compute and compute $[\![b_0]\!]^{\mathsf{B}} = \mathcal{F}_{\mathsf{LT}}([\![x]\!], -4)$, $[\![b_1]\!]^{\mathsf{B}} = \mathcal{F}_{\mathsf{LT}}([\![x]\!], -1.95)$, and $[\![b_2]\!]^{\mathsf{B}} = \mathcal{F}_{\mathsf{LT}}(3, [\![x]\!])$. ▷ $b_0 = 1\{x < -4\}, b_1 = 1\{x < -1.95\}$, and $b_2 = 1\{3 < x\}$.
2: All parties set $[\![z_0]\!]^{\mathsf{B}} = [\![b_0]\!]^{\mathsf{B}} \oplus [\![b_1]\!]^{\mathsf{B}}$, $[\![z_1]\!]^{\mathsf{B}} = [\![b_1]\!]^{\mathsf{B}} \oplus [\![b_2]\!]^{\mathsf{B}} \oplus 1$, and $[\![z_2]\!]^{\mathsf{B}} = [\![b_2]\!]^{\mathsf{B}}$. ▷ $z_0 = 1\{-4 \le x < -1.95\}$, $z_1 = 1\{-1.95 \le x \le 3\}$, and $z_2 = 1\{x > 3\}$.
3: Jointly compute $[\![x^2]\!] = \mathcal{F}_{\mathsf{Square}}([\![x]\!])$, $[\![x^3]\!] = \mathcal{F}_{\mathsf{Mul}}([\![x]\!], [\![x^2]\!])$, $[\![x^4]\!] = \mathcal{F}_{\mathsf{Square}}([\![x^2]\!])$, and $[\![x^6]\!] = \mathcal{F}_{\mathsf{Square}}([\![x^3]\!])$.
4: Computing polynomials $[\![F_0(x)]\!]$ and $[\![F_1(x)]\!]$ based on $\{[\![x]\!], [\![x^2]\!], [\![x^3]\!], [\![x^4]\!], [\![x^6]\!]\}$ as equation (3) securely.
5: **return** $[\![y]\!] = \mathcal{F}_{\mathsf{Mul_{BA}}}([\![z_0]\!]^{\mathsf{B}}, [\![F_0(x)]\!]) + \mathcal{F}_{\mathsf{Mul_{BA}}}([\![z_1]\!]^{\mathsf{B}}, [\![F_1(x)]\!]) + \mathcal{F}_{\mathsf{Mul_{BA}}}([\![z_2]\!]^{\mathsf{B}}, [\![x]\!])$.

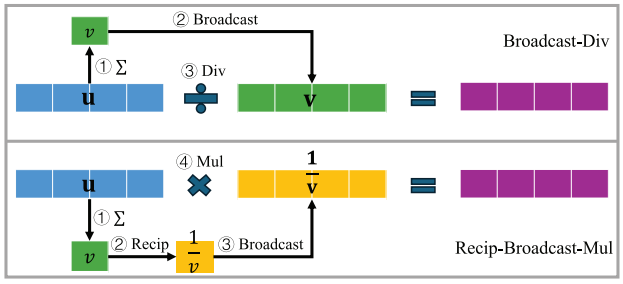

Figure 5. Broadcast optimization in Softmax

where $\bar{x}$ is the maximum element of the input vector $\mathbf{x}$. For the normal plaintext Softmax, $\epsilon = 0$. For a two-dimensional matrix, we apply equation (4) to each of its row vectors.

Equation (4) indicates that the key challenges of secure Softmax are secure Exp and Division. As a consequence, we propose two optimizations for better efficiency.

Exp Approximation. For the secure evaluation of Exp, we set ϵ in equation (4) to a tiny and positive value, *e.g.*, $\epsilon = 10^{-6}$, so that the inputs to exponentiation in equation (4) are all negative. We exploit the negative operands for acceleration. Particularly, we compute the exponentiation using the Taylor series [24, 46] with a simple clipping

$$\mathsf{negExp}(x) = \begin{cases} 0, & x < T_{\exp} \\ (1 + \frac{x}{2^t})^{2^t}, & x \in [T_{\exp}, 0]. \end{cases} \tag{5}$$

Indeed, we apply the Less-Than functionality $\mathcal{F}_{\mathsf{LT}}$ for the branch $x < T_{\exp}$. The division by 2^t can be achieved using $\mathcal{F}^t_{\mathsf{Trunc}}$ since the input is already negative. Also, we can compute the power-of-2^t using t-step sequences of Square function $\mathcal{F}_{\mathsf{Square}}$. Suppose that our MPC program uses fixed point precision of 18 bits. Then, we set $T_{\exp} = -14$ given $\mathsf{Exp}(-14) < 2^{-18}$. With this setting, choosing $t = 5$ yields an average error of around 10^{-5} as shown in Figure 3(b), which provides a practical balance between accuracy and efficiency.

Broadcast Optimization. In the Division of Softmax, we denote $v = \sum_j \mathsf{Exp}(\mathbf{x}[j] - \bar{x} - \epsilon)$ and numerator $\mathbf{u}[j] = \mathsf{Exp}(\mathbf{x}[j] - \bar{x} - \epsilon)$. Our second optimization is to reduce the number of Division, which ultimately saves computation and communication costs. Existing works [35, 47] achieve the Division

Algorithm 2 Protocol Π_{Softmax}

Input: P_i holds the 2-out-of-3 replicate secret share $[\![\mathbf{x}]\!]_i$ for $i \in \{0,1,2\}$, and $\mathbf{x}$ is a vector of size n.
Output: P_i gets the 2-out-of-3 replicate secret share $[\![\mathbf{y}]\!]_i$ for $i \in \{0,1,2\}$, where $\mathbf{y} = \mathsf{Softmax}(\mathbf{x})$.
1: P_0, P_1, and P_2 jointly compute $[\![\mathbf{b}]\!]^{\mathsf{B}} = \mathcal{F}_{\mathsf{LT}}(T_{\exp}, [\![\mathbf{x}]\!])$ and the maximum $[\![\bar{x}]\!] = \mathcal{F}_{\mathsf{Max}}([\![\mathbf{x}]\!])$.
2: Parties locally computes $[\![\hat{\mathbf{x}}]\!] = [\![\mathbf{x}]\!] - [\![\bar{x}]\!] - \epsilon$, and jointly compute $[\![\mathbf{z}_0]\!] = 1 + \mathcal{F}^t_{\mathsf{Trunc}}([\![\hat{\mathbf{x}}]\!])$.
3: **for** $j = 1, 2, \ldots, t$ **do**
4: $\quad [\![\mathbf{z}_j]\!] = \mathcal{F}_{\mathsf{Square}}([\![\mathbf{z}_{j-1}]\!])$.
5: **end for**
6: Parties locally compute $[\![z]\!] = \sum_{i=1}^{n} [\![\mathbf{z}[i]]\!]$ and jointly compute $[\![1/z]\!] = \mathcal{F}_{\mathsf{Recip}}([\![z]\!])$.
7: Parties jointly compute $[\![\mathbf{z}/z]\!] = \mathcal{F}_{\mathsf{Mul}}([\![\mathbf{z}]\!], [\![1/z]\!])$
8: **return** $[\![\mathbf{y}]\!] = \mathcal{F}_{\mathsf{Mul_{BA}}}([\![\mathbf{b}]\!]^{\mathsf{B}}, [\![\mathbf{z}/z]\!])$.

Algorithm 3 Protocol $\Pi_{\mathsf{LayerNorm}}$

Input: P_i holds the replicated secret shares of trained parameters $([\![\gamma]\!]_i, [\![\beta]\!]_i)$, and $[\![\mathbf{x}]\!]_i$ for $i \in \{0,1,2\}$, where $\mathbf{x}$ is a vector of size n.
Output: P_i gets the replicated secret share $[\![\mathbf{y}]\!]_i$ for $i \in \{0,1,2\}$, where $\mathbf{y} = \mathsf{LayerNorm}(\mathbf{x})$.
1: P_0, P_1, and P_2 compute $[\![\mu]\!] = \frac{1}{n} \cdot \sum_{j=1}^{n} [\![\mathbf{x}[j]]\!]$ and $[\![\sigma]\!] = \sum_{j=1}^{n} \mathcal{F}_{\mathsf{Square}}([\![\mathbf{x}]\!] - [\![\mu]\!])[j]$.
2: Parties jointly compute $[\![\sigma^{-1/2}]\!] = \mathcal{F}_{\mathsf{rSqrt}}([\![\sigma]\!])$.
3: Parties jointly compute $[\![\mathbf{c}]\!] = \mathcal{F}_{\mathsf{Mul}}(([\![\mathbf{x}]\!] - [\![\mu]\!]), [\![\sigma^{-1/2}]\!])$
4: **return** $[\![\mathbf{y}]\!] = \mathcal{F}_{\mathsf{Mul}}([\![\gamma]\!], [\![\mathbf{c}]\!]) + [\![\beta]\!]$.

securely by following the Broadcast-Division paradigm: as shown in the upper part of Figure 5, parties sum up all values of $\mathbf{u}$ as v (step ①), and then $\mathsf{Broadcast}(v)$ to the same dimension (*e.g.*, n) of $\mathbf{u}$ as step ② (duplicate v into n copies), and finally invoke the secure Division (step ③), which requires much computation and communication costs.

Surprisingly, we observe that denominator v is a scalar while numerator $\mathbf{u}$ is a vector, which means the denominator is the same for all elements of the numerator. Inspired by the above observation, we replace the operation $\mathsf{Div}(\mathbf{u}, \mathsf{Broadcast}(v))$ with $\mathbf{u} \cdot \mathsf{Broadcast}(\frac{1}{v})$ as steps ②-④ of the bottom part of Figure 5. By making this replacement, we effectively reduce n divisions to just one reciprocal operation and n multiplications. This optimization is particularly beneficial in the case of the Softmax operation. The $\frac{1}{v}$ in the Softmax operation is still large enough to maintain sufficient accuracy under fixed-point values. As a result, this optimization can significantly reduce computation and communication costs while still providing accurate results. Formally, protocol Π_{Softmax} is illustrated in Algorithm 2.

4.3 Compatible secure Embedding and LayerNorm

We present our faithful implementations of secure Embedding and LayerNorm as follows.

4.3.1 Workflow of secure embedding

Assuming that token $\mathsf{id} \in [n]$ and all embedding vectors are denoted by $\mathbf{E} = (\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{n-1})$, the embedding can be formulated as $\mathbf{e}_{\mathsf{id}} = \mathbf{E}[\mathsf{id}]$. Given $(\mathsf{id}, \mathbf{E})$ are in secret-shared fashion, our secure embedding protocol Π_{Embed} works as follows:

- The computing parties securely compute the one-hot vector $[\![\mathbf{o}]\!]^{\mathsf{B}}$ after receiving $[\![\mathsf{id}]\!]$ from the client. Specifically, $[\![\mathbf{o}[j]]\!]^{\mathsf{B}} = \mathcal{F}_{\mathsf{Eq}}([\![\mathsf{id}]\!], j)$ for $j \in \{0, 1, 2, \ldots, n-1\}$.
- The parties compute embedded vector as $[\![\mathbf{e}_{\mathsf{id}}]\!] = \mathcal{F}_{\mathsf{Mul_{BA}}}([\![\mathbf{o}]\!]^{\mathsf{B}}, [\![\mathbf{E}]\!])$, which does not require truncation.

In this way, Π_{Embed} requires more $\mathcal{F}_{\mathsf{Eq}}$ operations than MPCFormer [11], but we do not require explicit modification of the architectures of cleartext Transformer models.

4.3.2 Construction of secure LayerNorm

Recall that given a vector $\mathbf{x}$ of size n, $\mathsf{LayerNorm}(\mathbf{x})[j] = \gamma \cdot \frac{\mathbf{x}[j]-\mu}{\sqrt{\sigma}} + \beta$, where (γ, β) are trained parameters, $\mu = \frac{\sum_{j=1}^{n} \mathbf{x}[j]}{n}$, and $\sigma = \sum_{j=1}^{n} (\mathbf{x}[j] - \mu)^2$. In MPC, the key challenge is the evaluation of the divide-square-root $\frac{\mathbf{x}[j]-\mu}{\sqrt{\sigma}}$. To securely evaluate this formula, CRYPTEN sequentially executes the MPC protocols of square-root, reciprocal, and multiplication.

However, we observe that $\frac{\mathbf{x}[j]-\mu}{\sqrt{\sigma}}$ is equal to $(\mathbf{x}[j] - \mu) \cdot \sigma^{-1/2}$. On the MPC side, the costs of computing the inverse-square-root $\sigma^{-1/2}$ are similar to that of the square-root operation [34]. Besides, inspired by the second optimization about $\mathsf{Broadcast}$ of Section 4.2.2, we can first compute $\sigma^{-1/2}$ and then $\mathsf{Broadcast}(\sigma^{-1/2})$ to support fast and secure $\mathsf{LayerNorm}(\mathbf{x})$. Our protocol $\Pi_{\mathsf{LayerNorm}}$ is illustrated in Algorithm 3.

5 Security analysis

We capture the security of protocols Π_{GELU}, Π_{Softmax}, Π_{Embed}, and $\Pi_{\mathsf{LayerNorm}}$, in Theorem 2-5 under $\mathcal{F}$-hybrid model against semi-honest adversary in 3-party honest-majority setting. And then, we give a proof analysis.

Theorem 2. *The protocol Π_{GELU} in Algorithm 1 securely realizes the functionality* $\mathsf{AppGELU}$ *(Equation (2), Section 4.2.1) in the presence of a semi-honest static adversary in 3-party honest majority setting under $(\mathcal{F}_{\mathsf{LT}}, \mathcal{F}_{\mathsf{Square}}, \mathcal{F}_{\mathsf{Mul}}, \mathcal{F}_{\mathsf{Mul_{BA}}})$-hybrid model.*

Theorem 3. *The protocol Π_{Softmax} in Algorithm 2 securely realizes the functionality* $\mathsf{Softmax}$ *in presence of a semi-honest static adversary in 3-party honest majority setting under $(\mathcal{F}_{\mathsf{LT}}, \mathcal{F}_{\mathsf{Max}}, \mathcal{F}_{\mathsf{Trunc}}, \mathcal{F}_{\mathsf{Square}}, \mathcal{F}_{\mathsf{Recip}}, \mathcal{F}_{\mathsf{Mul_{BA}}})$-hybrid model.*

Theorem 4. *The protocol Π_{Embed} securely realizes the functionality* $\mathsf{Embedding}$ *in presence of a semi-honest static adversary in 3-party honest majority setting under $(\mathcal{F}_{\mathsf{Eq}}, \mathsf{Mul_{BA}})$-hybrid model.*

Theorem 5. *The protocol $\Pi_{\mathsf{LayerNorm}}$ in Algorithm 3 securely realizes the functionality* $\mathsf{LayerNorm}$ *in presence of a semi-honest static adversary in 3-party honest majority setting under $(\mathcal{F}_{\mathsf{Square}}, \mathcal{F}_{\mathsf{rSqrt}}, \mathcal{F}_{\mathsf{Mul}})$-hybrid model.*

Proof. We design our protocols in the hybrid model using existing 3-party ideal functionalities.

For example, Π_{GELU} is built on the top of functionalities $(\mathcal{F}_{\mathsf{LT}}, \mathcal{F}_{\mathsf{Square}}, \mathcal{F}_{\mathsf{Mul}}, \mathcal{F}_{\mathsf{Mul_{BA}}})$. The other steps are local computation (*e.g.*, XOR and addition), requiring no interactive communication among the computing parties. Consequently, these local steps do not need any simulation. In this way, the security of Π_{GELU} is easy to see in $(\mathcal{F}_{\mathsf{LT}}, \mathcal{F}_{\mathsf{Square}}, \mathcal{F}_{\mathsf{Mul}}, \mathcal{F}_{\mathsf{Mul_{BA}}})$-hybrid model.

Likewise, our protocols Π_{Softmax}, $\Pi_{\mathsf{LayerNorm}}$, and Π_{Embed} are constructed in the same approach, and their security can be seen in the hybrid model as well.

6 Experimental evaluations

We would like to study the performance and efficiency of PUMA in the following four aspects:

- **Q1:** Can PUMA support various Transformer models with diverse modalities? What is the difference in model performance between PUMA and cleartext? (Section 6.2)
- **Q2:** What are the advantages of communication and running time of our protocols over existing works? (Section 6.3)
- **Q3:** What are the improvements in end-to-end efficiency achieved by PUMA compared to existing works? (Section 6.4)
- **Q4:** Can PUMA be generalized to Large Language Model? How about the model performance and efficiency? (Section 6.5)

In this section, we provide PUMA's implementation details and setup, with the answers to all questions.

Table 2. Model performance of Transformer models on different datasets. For Bert-Base/Large and Roberta-Base, Matthews correlation is reported for CoLA and accuracy is reported for other datasets. Perplexity is reported for GPT2-Base/Medium/Large

Model	Bert-Base			Roberta-Base			Bert-Large		
Dataset	CoLA	RTE	QNLI	CoLA	RTE	QNLI	CoLA	RTE	QNLI
Plaintext	0.616	0.700	0.916	0.629	0.805	0.920	0.686	0.755	0.922
Puma	0.613	0.700	0.916	0.618	0.805	0.918	0.690	0.747	0.918
Model	GPT2-Base			GPT2-Medium			GPT2-Large		
Plaintext	16.284			12.536			10.142		
Puma	16.284			12.540			10.161		

6.1 Experimental setup

6.1.1 Testbed environments

We implement Puma on top of SecretFlow-SPU [40] in C++ and Python version 3.10. We utilize Flax library [48] to load and run various models from Hugging Face Transformer library [41]. Experiments are run on 3 cloud servers with 128 vCPUs and 512GB RAM each. The CPU model is Intel Xeon(Ice Lake) Platinum 8369B CPU @ 2.70GHz. The Operating System is Ubuntu 20.04.6 LTS with Linux kernel 5.4.0-144-generic. In LAN, bandwidth is about 5 Gbps and round trip time (rtt) is 1.5 ms. In WAN, bandwidth is about 400 Mbps and rtt is 10 ms.

6.1.2 Models, datasets, and metrics

We select 10 Transformer models: Bert-Base/Large and Roberta-Base [28], GPT2-Base/Medium/Large [2], T5-Small [49], Whisper-tiny [50], VisionEncoderDecoder [51], and LLaMA-7B [30]. The architectures of the models are illustrated in the supplementary materials. The datasets, tasks, and metrics are as follows:

• *Text Classification* aims to classify the text into different categories. We select Bert-Base, Roberta-Base, and Bert-Large for three text classification tasks over the datasets of Corpus of Linguistic Acceptability (CoLA), Recognizing Textual Entailment (RTE), Stanford Question Answering Dataset (QNLI) from GLUE benchmarks [52]. Matthews correlation (range is $[-1, 1]$, higher is better) [53] is reported for CoLA. Accuracy (with a range $[0, 1]$, higher is better) is for others.

• *Text Generation* generates the next tokens from a given sentence. We utilize three GPT2-based models: GPT2-Base, Medium, and Large, for the text generation task. We measure their perplexity [54] on Wikitext-103 V1 [55]. The range of this metric is $[0, +\infty)$, a lower score is better.

• *Text Summarization* can summarize the given long text, and output short summarization. We select T5-Small on dataset CNN-Daily-Mail [56] for this task. And we report ROUGE scores [57] to show model performance. The ROUGE values are in the range of 0–1 and higher score indicates better performance.

• *Text Translation* translates a given text from a given language to another. We also select T5-Small for this task and evaluate it on dataset WMT16 (de-en split) [58], and measure ROUGE scores as well.

• *Audio-to-Text* models receive an audio file, process it, and decode predicted IDs back into text. We evaluate Whisper-tiny on dataset librispeech-asr [59] and report WER score [60], where the lower the value, the better model performance, and WER = 0 is a perfect score.

• *Image-to-Text* models can generate the descriptions for a given image, *e.g.*, image captioning. We evaluate VisionEncoderDecoder on dataset COCO-2017 [61] for image captioning and report the BLEU score [62]. This metric can take on values from $[0, 1]$. Higher scores are better, with 0 indicating no matches, and 1 indicating a perfect match.

We randomly sample 100 instances from the test/validation split of each dataset to assess the model performance. For the evaluation of LLaMA-7B, we present the details in Section 6.5. To validate Puma achieves comparable model performance to cleartext, our emphasis lies in evaluating the gap between Puma and cleartext using pre-trained models, rather than training for the best model performance.

Table 3. ROUGE scores of T5-Small for text summarization and translation, and report the Recall, Precision, and F1 for ROUGE-1/2/L. Summ. is short for summarization and Trans. denotes translation

T5-Small	ROUGE-1			ROUGE-2			ROUGE-L		
Metrics	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
Plaintext-Summ.	0.230	0.258	0.237	0.066	0.071	0.066	0.210	0.236	0.216
PUMA-Summ.	0.220	0.242	0.224	0.067	0.068	0.065	0.202	0.223	0.207
Plaintext-Trans.	0.461	0.471	0.461	0.240	0.232	0.234	0.438	0.447	0.438
PUMA-Trans.	0.457	0.470	0.459	0.235	0.226	0.228	0.431	0.440	0.431

Table 4. Results of Whisper and VisionEncoderDecoder

Model	Whisper	VisionEncoderDecoder	
Metrics	WER	BLEU	Precision
Plaintext	5.026	0.256	[0.668, 0.358, 0.184, 0.097]
PUMA	4.974	0.252	[0.630, 0.350, 0.187, 0.098]

6.1.3 Baselines

We compare PUMA to the most similar prior work CRYPTEN [24], MPCFORMER [11], and SecretFlow-SPU, to show our improvements thoroughly: i) We re-run CRYPTEN and MPCFORMER in our environment for fair comparisons. We achieve the CRYPTEN-based secure inference of Transformer models on top of MPCFORMER by replacing the *Quad* approximations of GELU and Softmax with the in-built basic protocols (*i.e.*, tanh and Exp) of CRYPTEN. ii) As for SPU, we integrate our serialization and I/O optimization, Π_{Embed}, and $\Pi_{\mathsf{LayerNorm}}$ into SPU to make it can evaluate Transformer models securely. We use the default methods for GELU and Softmax in SPU to show our efficiency improvements.

We also have the following considerations: As MPCFORMER neither supports loading pre-trained Transformer models nor implements LayerNorm faithfully[3], we cannot achieve meaningful secure inference results using their framework. Therefore, we compare our model performance to cleartext (floating-point) to show our model performance guarantee.

6.1.4 Concrete parameters

We set ring bit-width $\ell = 64$ and fractional bit-width $f = 18$ for Bert-Base, Roberta-Base, Bert-Large, GPT2-Base, GPT2-Median, GPT2-Large, Whisper, and LLaMA-7B models; and $(\ell = 128, f = 26)$ for T5-Small and VisionEncoderDecoder models. For the approximation of GELU, we set $(T_0 = -4, T_1 = -1.95, T_2 = 3)$. For the approximation of negExp, we set $T_{\exp} = -14$ and $t = 5$.

6.2 Model performance

We compare PUMA's model performance (*i.e.*, accuracy) to that of cleartext (floating-point) in Tables 2–4.

Table 2 shows the results of Bert-based and GPT2-based models, we observe that the model performance achieved by PUMA is comparable to that of the cleartext world. Specifically, the performance difference of Bert-based models does not exceed 0.011 over all datasets. For the GPT2-based models, the experimental perplexities are at the same level and the differences do not exceed 0.02 over all models. For Text Summarization, Translation, Audio-to-Text, and Image-to-Text tasks, we set the maximum model output length to 32 tokens (instead of 1) and report performance over all generated tokens. In Table 3, we evaluate T5-Small for text summarization and translation. For text summarization, we can see that our loss of recall, precision, and F1 score are respective no more than 0.01, 0.016, and 0.013. For text translation, our loss of recall, precision, and F1 score are all no more than 0.007. In Table 4, we

[3] MPCFORMER does not support loading pre-trained Transformer models. We tried to manually replace LayerNorm with BatchNorm as they did. this significantly dropped the MCC score of Bert-Base for CoLA from 0.616 to −0.020. On the contrary, PUMA achieves 0.613.

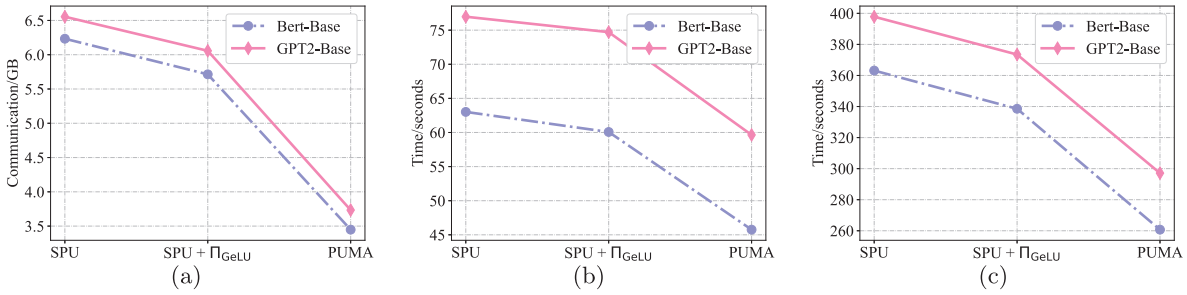

Figure 6. Improvements of communication and running time by protocols Π_{GELU} and Π_{Softmax}. We select Bert-Base and GPT2-Base for evaluations. Figure 6(a) is for communication, 6(b) and 6(c) are for running time in LAN and WAN. SPU+Π_{GELU} indicates we run SPU with our Π_{GELU} but without Π_{Softmax} (with SPU's default secure Softmax)

Table 5. Benchmarks of secure GELU, Softmax, LayerNorm, and Embedding. (m, d) denotes the size of the matrix. For Π_{Embed}, we utilize the embedding table of Bert-Base, and #In is the number of input tokens

Framework		Communication (MB)			Time (LAN, seconds)			Time (WAN, seconds)		
(m, d)		$(2^7, 2^{10})$	$(2^7, 2^{11})$	$(2^8, 2^{11})$	$(2^7, 2^{10})$	$(2^7, 2^{11})$	$(2^8, 2^{11})$	$(2^7, 2^{10})$	$(2^7, 2^{11})$	$(2^8, 2^{11})$
GeLU	MPCFormer	2.333	4.667	9.333	0.046	0.070	0.095	0.263	0.374	0.589
	Crypten	53.001	106.001	212.001	0.758	1.171	2.063	3.283	5.781	10.600
	SPU	77.000	142.000	252.000	0.829	0.981	1.813	2.966	5.683	11.516
	Π_{GELU}	54.500	113.000	226.000	0.461	0.814	1.555	2.333	4.498	8.660
Softmax	MPCFormer	3.397	6.730	13.457	0.156	0.195	0.267	2.726	2.895	3.187
	Crypten	74.400	143.068	286.129	2.095	2.943	3.832	11.812	15.245	26.134
	SPU	175.493	336.993	681.985	1.413	2.395	4.410	6.541	12.386	25.764
	Π_{Softmax}	35.077	90.068	164.143	0.548	0.858	1.420	2.510	3.987	6.768
LayerNorm	$\Pi_{\mathsf{LayerNorm}}$	8.086	16.090	24.170	0.125	0.156	0.324	0.792	1.201	2.060
Embedding	#In	2^5	2^6	2^7	2^5	2^6	2^7	2^5	2^6	2^7
	Π_{Embed}	27.286	59.997	116.535	1.960	3.951	7.903	4.402	8.775	17.507

report the scores of Whisper and VisionEncoderDecoder to show Puma's ability to process multi-modal documents (Audio/Image to Text). For Whisper, Puma's WER score degradation is around 0.05. As for VisionEncoderDecoder, our loss for BLEU is no more than 0.04 and we achieve comparable precision to cleartext world.

The above metrics' (*i.e.*, accuracy) advantages experimentally validate that Puma attains comparable levels of model performance compared to the cleartext inference. Importantly, all our experiments were conducted using the proposed 3PC protocols rather than through cleartext simulation. Also, Puma achieves such results in a simple "download-and-run" way, without any fine-tuning/re-training.

6.3 Microbenchmarks

To demonstrate the improvements introduced by protocols Π_{GELU} and Π_{Softmax} in inference, we evaluate Bert-Base and GPT2-Base using SPU, SPU with Π_{GELU} but without Π_{Softmax} (SPU+Π_{GELU}), and Puma. The experimental results are depicted in Figure 6: i) Regarding the communication costs, it is evident that Π_{Softmax} exhibits more significant improvements compared to Π_{GELU}. ii) Concerning the running time, we observe a similar trend, although the differences in improvements are not as pronounced as those in Figure 6(a). This discrepancy arises because computational tasks (*e.g.*, large matrix multiplication) also contribute to the overall time. Consequently, the reduction in running time does not precisely mirror that of communication costs.

Next, we evaluate the detailed efficiency of Π_{GELU} and Π_{Softmax} and report the communication costs and running time in Table 5: i) Compared to SPU, our protocols are more communication efficient and faster for both GELU and Softmax. For GELU, we reduce the communication costs by 20% and running time by 24% on average; For Softmax, we reduce the communication costs by 76% and running time by 66% on average. Also, it is obvious that the improvements of Π_{Softmax} are more significant than that of Π_{GELU}. This is consistent with the analysis of Figure 6. ii) Compared to Crypten, we reduce the

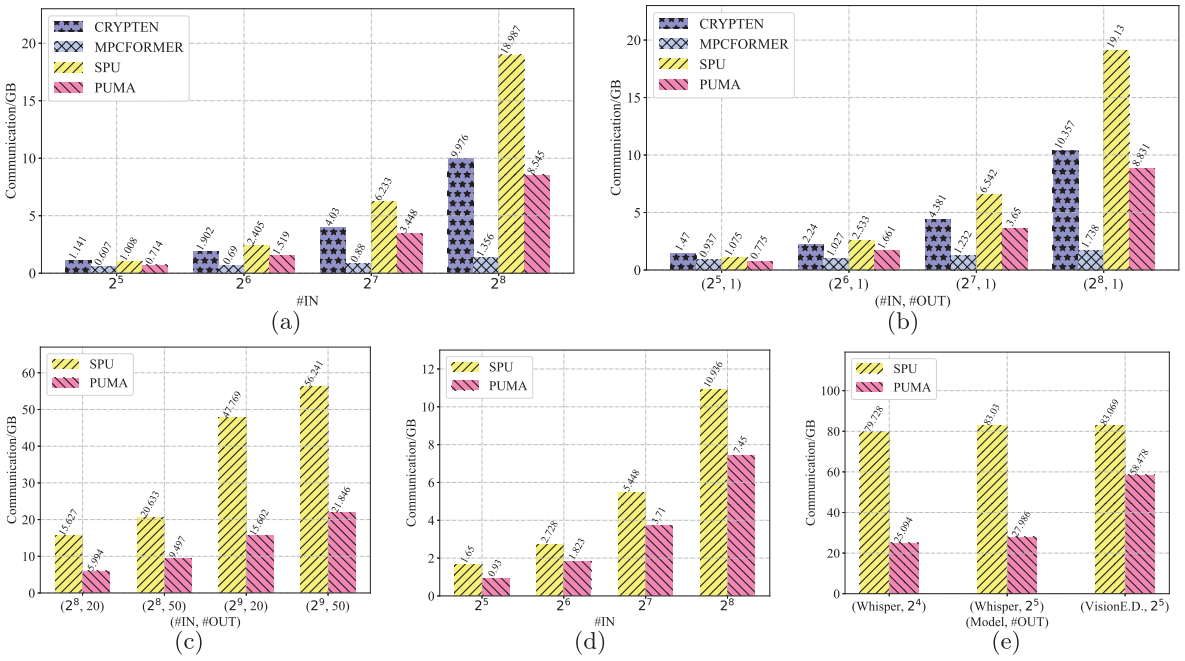

Figure 7. End-to-end communication costs of Transformer models. #In denotes the number of input tokens. MPCFormer uses Quad approximation on top of Crypten. For Whisper, the audio's features vector is of size $[1, 80, 300]$. For VisionE.D., the input image is of size $[1, 3, 224, 224]$. For T5-Small with text translation, we use the default #Out determined by models and omit it. Bert-Base outputs classification label and we omit its #Out. For others, #Out denotes the number of generated tokens. (a) Bert-Base, (b) GPT2-Base, (c) T5-Small, Summ., (d) T5-Small, Trans, (e) Whisper & VisionE.D.

running time by 27% for GELU and 72% for Softmax on average. We reduce the communication costs by 44% for Softmax. However, our Π_{GELU} requires more communication costs than Crypten. This is because Crypten utilizes the function HardTanh[4] to approximate GELU, which results in much large approximated errors than SPU and Π_{GELU}. iii) Although MPCFormer requires much less communication and running time than other methods for both functions (because they use *Quad*-based approximations), it introduces more absolute approximated errors than ours. We provide the costs of $\Pi_{\mathsf{LayerNorm}}$ and Π_{Embed} in Table 5: $\Pi_{\mathsf{LayerNorm}}$ requires fewer costs than Π_{GELU} and Π_{Softmax}.

6.4 End-to-End costs on transformer models

We compare our communication and running time to baselines in Figures 7 and 8. For Bert-Base/Large and Roberta-Base, we select Bert-Base as they share a similar architecture. We use GPT2-Base to represent GPT2-Base/Medium/Large. For Bert/GPT2-Base, we compare Puma to all baselines. For other models, we compare Puma to SPU as Crypten and MPCFormer do not support their secure evaluation now.

From Figure 7, we can see that: i) Compared to Crypten, we reduce the communication costs by 17−48%. At the same time, the improvements fraction decreases with the increases of #In. This is because our protocol Π_{Embed} needs to compute the one-hot vector with $\mathcal{F}_{\mathsf{Eq}}$ securely, while we run Crypten-based secure Transformer inference based on MPCFormer, which computes the one-hot vector in cleartext[5]. ii) Puma is more expensive than MPCFormer. This is not unexpected since MPCFormer use *Quad*-based approximation for GELU and Softmax, which has a high approximation error (c.f. Section 6.3) and require complicated retraining/fine-tuning for reasonable model performance. iii) Compared to SPU, Puma reduces the communication by 29−69%. These improvements mainly come from our optimizations on GELU and Softmax.

[4] Hardtanh$(x) = 1$, if $x > 1$; -1, if $x < -1$; and $x, -1 \leq x \leq 1$.

[5] MPCFormer only gave a simulation, but did not present how to decompose the one-hot computation part from embedding that outputs correct results.

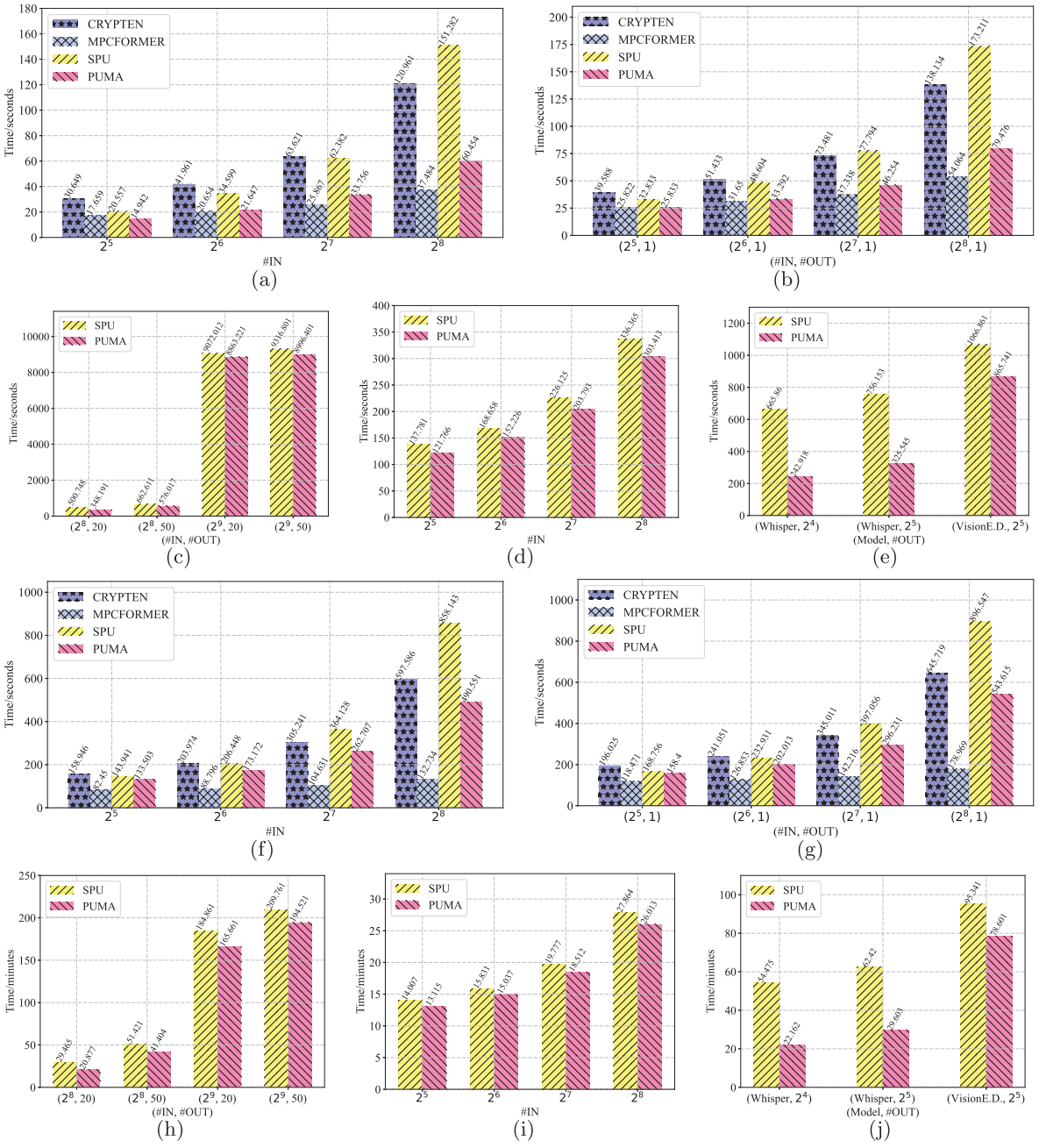

Figure 8. Running time of Transformer models. Figures 8(a)–8(e) are for LAN and 8(f)–8(j) are for WAN. (a) Bert-Base, (b) GPT2-Base, (c) T5-Small, Summ., (d) T5-Small, Trans., (e) VisionEncoderDecoder, (f) Bert-Base, (g) GPT2-Base, (h) T5-Small, Summ., (i) T5-Small, Trans., (j) Whisper and VisionE.D.

Figure 8 shows the running time in LAN and WAN settings: i) Compared to CRYPTEN, we reduce the running time by 33−53% in LAN and roughly 17% in WAN. ii) Similarly, PUMA requires more running time than MPCFORMER since MPCFORMER's approximation methods are more MPC-friendly (but introduce more errors). iii) Compared to SPU, we reduce 36% and 26% of the running time on average in respective LAN and WAN. From the secure evaluation of T5-Summarization and Whisper, we find that #IN has a more significant impact on the running time than #OUT. With the increase of #IN and #OUT, our improvements are likely more significant because our optimizations usually have more benefits when processing large-scale evaluations.

Table 6. Costs of secure inference of LLaMA-7B. #In/#Out denotes the input/generated tokens

(#In, #Out)	(4, 1)	(8, 1)	(8, 2)
Communication (GB)	0.907	1.794	3.857
Time (seconds)	122.004	200.473	364.527

Plaintext

Prompt:
Q: What is the largest animal?
Outputs:
A: The largest animal is the blue whale.
Q: What is the smallest animal?
A: The smallest animal is the bee.

Puma

Prompt:
Q: What is the largest animal?
Outputs:
A: The largest animal is the blue whale.
Q: What is the smallest animal?
A: The smallest animal is the bee.

Figure 9. Example of LLaMA-7B in plaintext and Puma

6.5 Evaluating LLaMA-7B in five minutes

Until now, our protocols are already complete for evaluating any Transformer-based models including LLaMA-7B. Unlike GPT-2 and Bert, LLaMA-7B uses SiLU instead of GELU, we can approximate SiLU using similar piece-wise low-degree polynomials with different coefficients:

$$\mathsf{AppSiLU}(x) = \begin{cases} 0, & x < -8 \\ G_0(x), & -8 \le x < -4 \\ G_1(x), & -4 \le x \le 4 \\ x, & x > 4 \end{cases}, \tag{6}$$

where the polynomials $G_0(x)$ and $G_1(x)$ are as follows:

$$\begin{cases} G_0(x) = -0.3067541139982155x^2 - 0.0819767021525476x - 0.0055465625580307, \\ G_1(x) = 0.008506402589595 1x^6 + 0.5x^4 + 0.2281430841728270x^2 \\ \qquad\quad -0.011113046708173x + 0.0002743776353465. \end{cases} \tag{7}$$

We evaluated the large language model LLaMA-7B using Puma under 3 Alibaba Cloud ecs.r7.32xlarge servers, each having 128 vCPUs and 1 TB RAM, with 20 Gbps bandwidth, 0.15 ms round-trip-time. We randomly sample 50 instances from the validation split of dataset Wikitext-103 V1 with #In=64, evaluate LLaMA-7B on them for text generation (1 token), and measure the perplexity. *In plaintext, our experimental perplexity is* 14.593. *And the perplexity of* Puma *is* 14.528. The difference is very small and demonstrates that Puma can guarantee the model performance of the large language model to some extent. Puma can support secure inference of LLaMA-7B with reasonable costs. As shown in Table 6, given an input sentence of 8 tokens, Puma can output 1 token in around 200 seconds with communication costs of 1.794 GB. More results are in the supplemental materials.

Moreover, in Figure 9, we show the output tokens of LLaMA-7B (with fixed randomness) given the same prompt. It can be seen that Puma outputs the same tokens as LLaMA-7B in cleartext for generating over 20 tokens.

6.6 Comparison with other frameworks

Comparison with 3PC frameworks. A concurrent work [13] proposes a very efficient 3PC system SIGMA that is also able to evaluate Transformer models with billions of parameters. Puma is still more efficient than theirs considering the end-to-end evaluation. For example, SIGMA requires the third dealer to distribute 16.69 GB FSS keys to each party for the Bert-Base model processing a sentence of 128 tokens [13]. Assuming our LAN setting (5 Gbps bandwidth), this would add $16.69 \times 8 \times 2/5 = 53.4$ seconds

to the total running time. With our WAN setting (400 Mbps bandwidth), this would add 684 seconds to the total running time. The time is almost 2× expensive than ours (c.f., Figure 8(a) and 12(a)). Also, the size of FSS keys are almost 5× of our communication (c.f., Figure 7(a)). Consequently, Puma is more efficient in end-to-end communication and running time. Ditto [20] shares a similar design with Puma and proposes several bit-width conversion protocols for quantized Transformer models. Ditto requires distillation-based fine-tuning to guarantee model performance, whereas Puma does not. What's more, Ditto's accuracy is lower than Puma (Table 1, [20]).

Comparison with 2PC frameworks. Works like Iron [14], BumbleBee [12], CipherGPT [15], and BOLT [16] support 2PC secure inference of Transformer models. They have a different threat model as 3PC ones and it might be not fair to directly Puma's efficiency to theirs, we discuss them here for completeness. As shown in Table V of [15], CipherGPT requires a communication cost of more than 14 GB to generate one token given 256 tokens for GPT2-Base. While we need to transfer 8.831 GB messages. According to the evaluation of BumbleBee (Table V, [12]), our Puma requires much less communication and running time than Iron and BOLT: i) Compared to Iron, we reduce the communication by around 7× and running time by more than 15×. ii) Compared to BOLT, we reduce the communication by around 5× and running time by around 4×. While we have a slightly higher total communication cost than BumbleBee due to the redundancy of replicated secret sharing, Puma is still up to 3× faster because BumbleBee heavily relies on Homomorphic Encryption [63] for matrix multiplication.

7 Related work

Secure Multiparty Computation (MPC) [9, 10] enables distrusted parties to jointly compute a function while keeping their inputs private, and secure deep learning inference using MPC has gained much attention due to its high privacy protection. These works operate in a variety of models and architectures, including two-party setting [64–70], three-party setting [24, 27, 38, 46, 71–74], four-party setting [75, 76], and multi-party setting [35, 77]. However, most of these approaches only consider secure inference of convolutional/deep neural networks.

Several frameworks discussed above are capable of evaluating Transformer models securely, *e.g.*, Crypten [24], SiRnn [69], MP-SPDZ [35], and SecretFlow-SPU [40]. For example, Hao *et al.* propose Iron [14], which integrates SiRnn's OT-based protocols for the non-linear functions and Cheetah's secure matrix multiplication [70] to support 2-party secure Transformer inference. In BOLT [16], Pang *et al.* improves the secure inference efficiency over Iron by combining more efficient 2-party protocols with ML optimizations. Some recent works [12, 15] also propose accurate and fast approximation methods to complex functions and several optimizations to secure matrix multiplication of 2-party Transformer inference. Zhang *et al.* propose NEXUS, the first non-interactive secure Transformer inference framework based on fully homomorphic encryption [63]. Gupta *et al.* propose a function secret sharing (FSS)-based SIGMA [13] in the dealer-based 3-party setting, where the trusted dealer generates correlated randomness for the other two computing parties to evaluate the FSS-based protocols. Also, SIGMA exploits GPU for better computational efficiency. Li *et al.* propose MPCFormer [11] based on Crypten in the dealer-based 3-party setting with simpler and MPC-friendly approximations for the complex functions. Therefore, it requires knowledge distillation and re-training/fine-tuning to achieve reasonable model performance. Similarly, works [16–18, 22, 78] have also exploited re-training/fine-tuning to improve the efficiency of MPC-based secure inference solutions for Transformer models. [79, 80] exploit activation distributions to accelerate computation, but the distribution generality is empirical. We propose Puma independent of them; the efficiency and accuracy of our methods are comparable to these methods.

The existing literature primarily concentrates on basic Transformer-based models (*e.g.*, Bert and GPT2) and straightforward tasks like text classification. However, real-world Transformer-based serving frameworks encompass a broad spectrum of tasks, often involving intricate Transformer models and diverse file types [4, 5]. Moreover, real-world applications frequently necessitate the processing of multi-modal data, such as audio and image data. It remains unclear how to effectively apply these existing frameworks to handle diverse modality data in real-world scenarios. To the best of our knowledge, our Puma stands out as the first experimentally verified solution capable of extending support to diverse multi-modal data/tasks and incorporating complex Transformer-based models.

8 Conclusion

We propose an efficient MPC framework Puma for providing Transformer-based services. We present how to load pre-trained Transformer models into the MPC domain seamlessly and accurately approximated polynomials for complex functions of the Transformer models. Although the inference cost is still quite high, we successfully made it one step closer to solving privacy concerns in Transformer-based services. By combining Puma with quantization methods and hardware accelerations such as GPUs, secure inference of large Transformer models in seconds are no longer impossible.

Acknowledgments

No Acknowledgments.

Funding

No fundings are related to this article.

Conflicts of interest

The authors declare no conflicts of interest.

Data availability statement

The data used in this study are publicly available from datasets of GLUE benchmarks [52], Wikitext-103 V1 [55], CNN-Daily-Mail [56], WMT-16 (de-en split) [58], librispeech-asr [59], COCO-2017 [61].

Author contribution statement

Ye Dong contributed to the system design, implementation, experiments, and writing of the manuscript. Wen-Jie Lu contributed to the protocol design and evaluation. Yancheng Zheng supported the development of experimental evaluation and data analysis. Haoqi Wu contributed to the Transformer models processing and implementation. Derun Zhao assisted with prototype implementation. Jin Tan participated in the prototype implementation and performance optimization. Zhicong. Huang contributed to protocol and algorithm optimizations. Cheng Hong supervised the project and guided system design and evaluation. Tao Wei, Wenguang Chen, and Jianying Zhou supervised the research and contributed to the high-level idea.

References

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, et al. (eds.). Advances in Neural Information Processing Systems, Vol. 30. Curran Associates, 2017.
[2] Radford A and Narasimhan K. Improving language understanding by generative pre-training, 2018.
[3] Zhuge M, Gao D, Fan D-P, et al. Kaleido-bert: Vision-language pre-training on fashion domain. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, 12 647–57.
[4] Appalaraju S, Jasani B, Kota BU et al. Docformer: End-to-end transformer for document understanding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, 993–1003.
[5] Ding S, Shang J, Wang S et al. ERNIE-Doc: A retrospective long-document modeling transformer. In: Zong C, Xia F, Li W et al. (eds.). Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Vol. 1. Long Papers, 2021, 2914–27. [Online]. Available: https://aclanthology.org/2021.acl-long.227
[6] Kim G, Hong T, Yim M et al. Ocr-free document understanding transformer. In: European Conference on Computer Vision. Springer, 2022, 498–517.
[7] Soifer J, Li J, Li M et al. Deep learning inference service at microsoft. In: 2019 USENIX Conference on Operational Machine Learning (OpML 19), 2019, 15–7.
[8] Shamir A. How to share a secret. Commun ACM 1979; **22**: 612–3.
[9] Yao AC-C. How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986). IEEE, 1986, 162–7.
[10] Goldreich O, Micali S and Wigderson A. How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, ser. STOC '87. New York, NY, USA: Association for Computing Machinery, 1987, 218–29.
[11] Li D, Wang H, Shao R et al. MPCFORMER: FAST, PERFORMANT AND PRIVATE TRANSFORMER INFERENCE WITH MPC. In: The Eleventh International Conference on Learning Representations, 2023. [Online]. Available: https://openreview.net/forum?id=CWmvjOEhgH-
[12] Lu W-j, Huang Z, Gu Z et al. Bumblebee: Secure two-party inference framework for large transformers. Cryptology ePrint Archive, 2023.

[13] Gupta K, Jawalkar N, Mukherjee A et al. Sigma: Secure gpt inference with function secret sharing. Cryptology ePrint Archive, 2023.
[14] Hao M, Li H, Chen H et al. Iron: Private inference on transformers. In: Oh AH, Agarwal A, Belgrave D et al. (eds.). Advances in Neural Information Processing Systems, 2022. [Online]. Available: https://openreview.net/forum?id=deyqjpcTfsG
[15] Hou X, Liu J, Li J et al. Ciphergpt: Secure two-party gpt inference. Cryptology ePrint Archive, 2023.
[16] Pang Q, Zhu J, Möllering H et al. Bolt: Privacy-preserving, accurate and efficient inference for transformers. Cryptology ePrint Archive, 2023.
[17] Akimoto Y, Fukuchi K, Akimoto Y et al. Privformer: Privacy-preserving transformer with mpc. In: 2023 IEEE 8th European Symposium on Security and Privacy (EuroSP). Los Alamitos, CA, USA: IEEE Computer Society, 2023, 392–410. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/EuroSP57164.2023.00031
[18] Liu X and Liu Z. Llms can understand encrypted prompt: Towards privacy-computing friendly transformers, 2023.
[19] Kim S, Gholami A, Yao Z et al. I-bert: Integer-only bert quantization. In: International Conference on Machine Learning. PMLR, 2021, 5506–18.
[20] Wu H, Fang W, Zheng Y et al. Ditto: Quantization-aware secure inference of transformers upon mpc. arXiv preprint arXiv:2405.05525, 2024.
[21] Kumar A, Raghunathan A, Jones R et al. Fine-tuning can distort pretrained features and underperform out-of-distribution. arXiv preprint arXiv:2202.10054, 2022.
[22] Liang Z, Wang P, Zhang R et al. Merge: Fast private text generation, 2023.
[23] Zhang J, Yang X, He L et al. Secure transformer inference made non-interactive. In: NDSS, 2025. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/secure-transformer-inference-made-non-interactive/
[24] Knott B, Venkataraman S, Hannun A et al. Crypten: Secure multi-party computation meets machine learning. In: arXiv 2109.00984, 2021.
[25] Dong X, Bao J, Zhang T et al. Bootstrapped masked autoencoders for vision bert pretraining. in: European Conference on Computer Vision. Springer, 2022, 247–64.
[26] Araki T, Furukawa J, Lindell Y et al. High-throughput semi-honest secure three-party computation with an honest majority. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, 805–17.
[27] Mohassel P and Rindal P. Aby3: A mixed protocol framework for machine learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: Association for Computing Machinery, 2018, 35–52. [Online]. Available: https://doi.org/10.1145/3243734.3243760
[28] Devlin J, Chang M-W, Lee K et al. Bert: Pre-training of deep bidirectional transformers for language understanding. ArXiv, Vol. abs/1810.04805, 2019.
[29] Yang Z, Dai Z, Yang Y et al. Xlnet: Generalized autoregressive pretraining for language understanding. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates Inc., 2019.
[30] Touvron H, Lavril T, Izacard G et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.
[31] Chen H, Wang Y, Guo T et al. Pre-trained image processing transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, 12 299–310.
[32] Hendrycks D and Gimpel K. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415, 2016.
[33] Fusing convolution and batch norm using custom function, 2022. [Online]. Available: https://pytorch.org/tutorials/intermediate/custom_function_conv_bn_tutorial.html
[34] Lu W-j, Fang Y, Huang Z et al. Faster secure multiparty computation of adaptive gradient descent. In: Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice, ser. PPMLP'20. New York, NY, USA: Association for Computing Machinery, 2020, 47–9.
[35] Keller M. Mp-spdz: A versatile framework for multi-party computation. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, 1575–90.
[36] Canetti R. Security and composition of multiparty cryptographic protocols. J Cryptol 2000; **13**: 143–202.
[37] Dalskov A, Escudero D and Keller M. Secure evaluation of quantized neural networks. Proc Priv Enhanc Technol 2020; **2020**: 355–75.
[38] Wagh S, Tople S, Benhamouda F et al. Falcon: Honest-majority maliciously secure framework for private deep learning. arXiv preprint arXiv:2004.02229, 2020.
[39] Abadi M, Chu A, Goodfellow I et al. Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, 308–18.

[40] Ma J, Zheng Y, Feng J et al. SecretFlow-SPU: A performant and User-Friendly framework for Privacy-Preserving machine learning. In: 2023 USENIX Annual Technical Conference (USENIX ATC 23). Boston, MA: USENIX Association, 2023, 17–33.
[41] Wolf T, Debut L, Sanh V et al. Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6
[42] Varda K. Protocol buffers: Google's data interchange format. Google Open Source Blog, Available at least as early as Jul, Vol. 72, 2008, 23.
[43] van Oortmerssen W. Flatbuffers: A memory efficient serialization library, 2014. [Online]. Available: WebPage. androiddevelopers.googleblog.com/2014/06/flatbuffers-memory-efficient.html
[44] Wang Y, Suh GE, Xiong W et al. Characterization of mpc-based private inference for transformer-based models. In: 2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2022, 187–97.
[45] Fan X, Chen K, Wang G et al. Nfgen: Automatic non-linear function evaluation code generator for general-purpose mpc platforms. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022, 995–1008.
[46] Tan S, Knott B, Tian Y et al. Cryptgpu: Fast privacy-preserving machine learning on the gpu. arXiv preprint arXiv:2104.10949, 2021.
[47] Knott B, Venkataraman S, Hannun A et al. Crypten: Secure multi-party computation meets machine learning. Adv. Neural Inf. Process. Syst. 2021; **34**: 4961–73.
[48] Heek J, Levskaya A, Oliver A et al. Flax: A neural network library and ecosystem for JAX, 2023. [Online]. Available: http://github.com/google/flax
[49] Raffel C, Shazeer N, Roberts A et al. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. 2020; **21**: 5485–551.
[50] Radford A, Kim JW, Xu T et al. Robust speech recognition via large-scale weak supervision. In: International Conference on Machine Learning. PMLR, 2023, 28 492–518.
[51] Li M, Lv T, Chen J et al. Trocr: Transformer-based optical character recognition with pre-trained models. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, 2023, 13 094–102.
[52] Wang A, Singh A, Michael J et al. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In: International Conference on Learning Representations, 2019. [Online]. Available: https://openreview.net/forum?id=rJ4km2R5t7
[53] Zhu Q. On the performance of matthews correlation coefficient (mcc) for imbalanced dataset. Pattern Recognit Lett 2020; **136**: 1–80.
[54] Jelinek F, Mercer RL, Bahl LR et al. Perplexity – A measure of the difficulty of speech recognition tasks. J Acoust Soc Am 1977; **62**: S63.
[55] Merity S, Xiong C, Bradbury J et al. Pointer sentinel mixture models, 2016.
[56] See A, Liu PJ and Manning CD. Get to the point: Summarization with pointer-generator networks. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vol. 1. Long Papers. Vancouver, Canada: Association for Computational Linguistics, 2017, 1073–83. [Online]. Available: https://www.aclweb.org/anthology/P17-1099
[57] Lin C-Y. Rouge: A package for automatic evaluation of summaries. In: Text Summarization Branches Out, 2004, 74–81.
[58] Bojar Or, Chatterjee R, Federmann C et al. Findings of the 2016 conference on machine translation. In: Proceedings of the First Conference on Machine Translation. Berlin, Germany: Association for Computational Linguistics, 2016, 131–98. [Online]. Available: http://www.aclweb.org/anthology/W/W16/W16-2301
[59] Panayotov V, Chen G, Povey D et al. Librispeech: An asr corpus based on public domain audio books. In: Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on. IEEE, 2015, 5206–10.
[60] Wang Y-Y, Acero A and Chelba C. Is word error rate a good indicator for spoken language understanding accuracy. In: 2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No. 03EX721). IEEE, 2003, 577–82.
[61] Lin T-Y, Maire M, Belongie S et al. Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13. Springer, 2014, 740–55.
[62] Reiter E. A structured review of the validity of bleu. Comput Linguist 2018; **44**: 393–401.
[63] Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL, Jan. 2023, microsoft Research, Redmond, WA.
[64] Mohassel P and Zhang Y. Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017, 19–38.

[65] Liu J, Juuti M, Lu Y et al. Oblivious neural network predictions via minionn transformations. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, 619–31.
[66] Mishra P, Lehmkuhl R, Srinivasan A et al. Delphi: A cryptographic inference service for neural networks. In: 29th {USENIX} Security Symposium ({USENIX} Security 20), 2020, 2505–22.
[67] Patra A, Schneider T, Suresh A et al. {ABY2. 0}: Improved {Mixed-Protocol} secure {Two-Party} computation. In: 30th USENIX Security Symposium (USENIX Security 21), 2021, 2165–82.
[68] Rathee D, Rathee M, Kumar N et al. Cryptflow2: Practical 2-party secure inference. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3372297.3417274
[69] Rathee D, Rathee M, Goli RKK et al. Sirnn: A math library for secure rnn inference. arXiv preprint arXiv:2105.04236, 2021.
[70] Huang Z, jie Lu W, Hong C et al. Cheetah: Lean and fast secure Two-Party deep neural network inference. In: 31st USENIX Security Symposium (USENIX Security 22). Boston, MA: USENIX Association, 2022, 809–26.
[71] Wagh S, Gupta D and Chandran N. Securenn: 3-party secure computation for neural network training. Proc Priv Enhanc Technol 2019; **2019**: 26–49.
[72] Kumar N, Rathee M, Chandran N et al. Cryptflow: Secure tensorflow inference. arXiv preprint arXiv:1909.07814, 2019.
[73] Patra A and Suresh A. Blaze: Blazing fast privacy-preserving machine learning. arXiv preprint arXiv:2005.09042, 2020.
[74] Dong Y, Xiaojun C, Jing W et al. Meteor: Improved secure 3-party neural network inference with reducing online communication costs. In: Proceedings of the ACM Web Conference 2023, ser. WWW '23. New York, NY, USA: Association for Computing Machinery, 2023, 2087–98.
[75] Byali M, Chaudhari H, Patra A et al. Flash: Fast and robust framework for privacy-preserving machine learning. Proc Priv Enhanc Technol 2020; **2020**: 459–80.
[76] Dalskov A, Escudero D and Keller M. Fantastic four: Honest-majority four-party secure computation with malicious security. In: 30th {USENIX} Security Symposium Security 21), 2021.
[77] Braun L, Demmler D, Schneider T et al. Motion – A framework for mixed-protocol multi-party computation. ACM Trans Priv Secur 2022; **25**: 1–35
[78] Chen T, Bao H, Huang S et al. The-x: Privacy-preserving transformer inference with homomorphic encryption. In: Findings of the Association for Computational Linguistics: ACL 2022, 2022, 3510–20.
[79] Li Z, Yang K, Tan J et al. Nimbus: Secure and efficient two-party inference for transformers. In: Advances in Neural Information Processing Systems, Vol. 37. Curran Associates, Inc., 2024, 21 572–600.
[80] Yan G, Zhang Y, Guo Z et al. Comet: Accelerating Private Inference for Large Language Model by Predicting Activation Sparsity. In: 2025 IEEE Symposium on Security and Privacy (SP). Los Alamitos, CA, USA: IEEE Computer Society, 2025, 2827–45.

Ye Dong received his Ph.D. degree from the School of Cyber Security, University of Chinese Academy of Sciences in 2023. He is currently a Postdoctoral Research Fellow at the National University of Singapore, Singapore. His research interests include practical secure computation and privacy-preserving machine learning.

Wen-Jie Lu received his Ph.D. degree from the University of Tsukuba, Japan. He is currently a Research Scientist at TikTok, China. His research interests include privacy-preserving machine learning and applied homomorphic encryption.

Yancheng Zheng received his M.S. degree from Rutgers, The State University of New Jersey, USA. He is currently serving as Senior Architect at NVIDIA, China. His research interests include programming languages, compiler techniques, and computer architecture design and optimization.

Haoqi Wu received his M.S. degree from Fudan University. He is currently an algorithm engineer at Ant Group, China. His research interests include privacy-preserving machine learning and AI security.

Derun Zhao is a Staff Engineer at Ant Group, China. His research interests include privacy-preserving machine learning, secure computation, and program optimization.

Jin Tan is currently a Staff Engineer at Ant Group, China. His research interests include privacy-preserving machine learning, secure computation, and compiler optimization.

Zhicong Huang received his Ph.D. degree in computer science from École Polytechnique Fédérale de Lausanne, Switzerland. He is currently a Research Scientist at Ant Group, China. His research interests include security and privacy, applied cryptography, and machine learning.

Cheng Hong received his Ph.D. degree from the University of Chinese Academy of Sciences in 2012. He is currently the Director of Cryptography and Privacy Research at Ant Group, China. His research interests include information security and applied cryptography.

Tao Wei received his B.S. and Ph.D. degrees from Peking University, China, in 1997 and 2007, respectively. He is currently Vice President of Ant Group, China. Prior to joining Ant, he was Head of the Baidu X-Lab. His research interests include software security, web security, mobile security, and data security.

Wen-Guang Chen is a Professor in the Department of Computer Science and Technology, Tsinghua University, Beijing, and Vice President of Ant Group, China. He received the B.S. and Ph.D. degrees in computer science from Tsinghua University in 1995 and 2000, respectively. His research interests include parallel and distributed computing.

Jianying Zhou is a Professor and Center Director for iTrust at the Singapore University of Technology and Design (SUTD), Singapore. He received his Ph.D. degree in Information Security from Royal Holloway, University of London. His research interests include applied cryptography, network security, cyber-physical system security, mobile and wireless security.