A Survey on Private Transformer Inference

YANG LI*, Nanyang Technological University, Singapore

XINYU ZHOU, Nanyang Technological University, Singapore

YITONG WANG, Nanyang Technological University, Singapore

LIANGXIN QIAN, Nanyang Technological University, Singapore

JUN ZHAO, Nanyang Technological University, Singapore

CCS Concepts: • **Computing Methodologies** → **Artificial intelligence**.

Additional Key Words and Phrases: Transformer models, data security, private inference services

1 INTRODUCTION

Transformer models have emerged as game-changers to revolutionize the field of Artificial Intelligence (AI). For instance, both ChatGPT [42] and Bing [40] have made the power of transformer-based models widely accessible, democratizing advanced AI capabilities. These models leverage attention mechanisms [55] adeptly to capture long-range dependencies in sequences of input tokens, allowing them to accurately model contextual information. Besides, unlike traditional task-specific learning approaches, large transformer models (e.g., GPT [46] and BERT [10]) are trained on huge quantities of unlabeled textual data and are directly useful for a wide variety of applications such as sentiment analysis, language translation, content generation, and question answering.

However, the application of large transformers still presents certain risks, particularly regarding privacy issues [35, 52]. Most popular transformer models operate in a pattern called Machine Learning as a Service (MLaaS), where a server provides the model and inference services to users who own the data. For instance, OpenAI provides ChatGPT as an online platform and offers remote APIs for developers, allowing users to access services by submitting prompts or messages. Nevertheless, this pattern raises privacy concerns: users need to transmit their private data to a company's server and have no direct control over how their data is handled. They must trust that the server processes the data honestly and follows the agreed terms of service. There exists a risk that the server could misuse the data, including unauthorized processing, storing the data indefinitely, or even selling it to third parties. Even if the server is trustworthy, there are inherent risks associated with centralized data storage, such as data breaches or unauthorized access by

Authors' Contact Information: Yang Li, yang048@@e.ntu.edu.sg, Nanyang Technological University, Singapore; Xinyu Zhou, Nanyang Technological University, Singapore, xinyu003@e.ntu.edu.sg; Yitong Wang, Nanyang Technological University, Singapore, yitong002@e.ntu.edu.sg; Liangxin Qian, Nanyang Technological University, Singapore, qian0080@e.ntu.edu.sg; Jun Zhao, Nanyang Technological University, Singapore, junzhao@ntu.edu.sg.

malicious insiders. These risks are particularly concerning when dealing with sensitive or personally identifiable information. Therefore, while MLaaS offers significant convenience and computational power, it also necessitates careful consideration of privacy issues. As a result, serious user data privacy concerns have been raised, which even led to a temporary ban of ChatGPT in Italy [33, 38]. Hence, there exists a gap between high-performance transformer inference services and privacy concerns, motivating the study of *Private Transformer Inference* (PTI).

Private inference is a cryptographic protocol that allows for model inference while ensuring that the server gains no knowledge about the users' input, and the users learn nothing about the server's model, apart from inference results.

Recently, private inference on transformers has been achieved by using private outsourced computation techniques, such as secure Multi-Party Computation (MPC) [61] and Homomorphic Encryption (HE) [15]. MPC enables multiple parties to jointly compute a function over their inputs while keeping those inputs private. Its essence is to allow the computation of results without any party revealing their private data to others. HE provides a way to perform computations on encrypted data without needing to decrypt it. This means that data can be processed in its encrypted form, preserving privacy and security throughout the computation process. HE is extremely useful in MLaaS as it allows clients to encrypt their data before sending it to the cloud for processing. The server can then compute the model's functions directly on the encrypted data, providing results without ever accessing the raw data.

In this paper, we review state-of-the-art papers that implement PTI using different private computation techniques, primarily focusing on MPC and HE technologies. We discuss their pros and cons and propose some future avenues to tackle. Furthermore, we propose a set of evaluation guidelines to compare solutions in terms of resource requirements. The key contributions of this paper are summarized as follows:

(1) Provide a comprehensive literature review of proposed solutions in the field of secure transformer inference from recent three years (2022-2024);

(2) Provide a breakdown of the main challenges faced in this field, as well as typical solutions that are implemented to mitigate them;

The rest of the survey is organized as follows: In Section 2, we introduce background information on the transformer architecture and cryptographic primitives. In Section 4, we provide an overview of the selected PTI studies, the approaches, strengths and weaknesses. Linear layers in transformers (i.e., large matrix multiplications) and the efficient computing protocols under HE mechanisms are discussed in Section 5. Non-linear layers (e.g., Softmax, GELU) in transformers and their corresponding efficient protocols are discussed in Section 6. In Section 7, we compare the experimental results presented by the selected papers and analyze them based on different metrics. In Section ??, we discuss challenges and future directions. Section 8 is the summary of the whole survey.

2 BACKGROUND

2.1 Private Inference

Consider a scenario where a client C possesses private data x, and a server S hosts a model M. The client C aims to utilize M to compute the inference result $M(x)$. The privacy requirement is that the server learns nothing about x, and the client learns nothing about M except what can be inferred from the $M(x)$. Following [64], we formally define private inference as follows:

DEFINITION 1. *A protocol* Π *between server* S *with model* M *and client* C *with input* x *is considered private if it satisfies:*

- ***Correctness.*** *The final output of protocol* Π*, denoted as* y*, is as same as the correct inference result* $M(x)$*.*

- **Security.** *Here we introduce the concept of Simulator* (Sim) *to define the protocol security. A simulator is a hypothetical algorithm that mimics the behavior of a party during protocol execution.*
  - **Corrupted client.** *Even if client C is corrupted, there still exists an efficient simulator* Sim_C *such that* $\mathrm{View}_C^{\Pi} \approx \mathrm{Sim}_C(x, y)$, *where* View_C^{Π} *denotes corrupted C's view during the execution of Π, i.e., an attacker on C cannot get more information through the execution than through the simulator.*
  - **Corrupted server.** *Even if server S is corrupted, there still exists an efficient simulator* Sim_S *such that* $\mathrm{View}_S^{\Pi} \approx \mathrm{Sim}_S(\mathcal{M})$, *where* View_S^{Π} *denotes corrupted S's view during the execution of Π.*

This definition provides a strong guarantee of privacy, ensuring that the inference process does not compromise data and model confidentiality. Please note that private inference does not consider attacks based on inference results (e.g., model inversion), and the protection from such attacks falls outside the scope of this study.

2.2 Transformer Architecture

Transformer is an encoder-decoder architecture in which both parts have a similar structure. We mainly focus on the encoder here. The encoder comprises a stack of identical blocks, each with two sub-layers: a multi-head self-attention mechanism and a feed-forward network. Residual connection and layer normalization (LayerNorm) are utilized around each of the two sub-layers. An encoder's architecture and process flow are shown in Fig. 1.

Fig. 1. Structure and workflow of a Transformer [64].

Embedding. At the start of the encoder, an embedding layer is employed to transform input tokens into continuous feature vector representations. Concretely, given $X_{\mathrm{input}} \in \mathbb{R}^{m \times 1}$, an embedding lookup table is used to generate output $X \in \mathbb{R}^{m \times d}$, where m denotes the length of tokens and d represents the model dimension.

Attention. Attention layers capture context and dependencies among the input through the multi-head attention mechanism. Specifically, an input $X \in \mathbb{R}^{m \times d}$ is fed into L multi-head attention layers. Each layer linearly projects the

input X using the query, key and value weights $W_h^Q, W_h^K, W_h^V \in \mathbb{R}^{d \times d'}$, where $h \in [H]$, $d' = d/H$ and H is the number of heads in each multi-head attention layer. The results of the projections are $Q_h, K_h, V_h \in \mathbb{R}^{m \times d'}$. Then, the attention mechanism works as the following function:

$$\text{Attention}(Q_h, K_h, V_h) = \text{Softmax}\left(\frac{Q_h K_h^T}{\sqrt{d}}\right) V_h, \tag{1}$$

where the output can be denoted as $X_h^{\text{Att}} \in \mathbb{R}^{m \times d'}$ for simplification. Furthermore, all attention results $[X_h^{\text{Att}} |_{h \in [H]}]$ are concatenated to generate the final output $X^{\text{Att}} \in \mathbb{R}^{m \times d}$.

Feed-Forward. A fully connected feed-forward layer consists of two linear transformations with a Gaussian Error Linear Unit (GeLU) activation in between, which can be formalized as follows:

$$\text{FeedForward}(X) = \text{GeLU}(X W_1 + b_1) W_2 + b_2. \tag{2}$$

LayerNorm. The LayerNorm is applied after the attention and feed-forward layers, ensuring that the inputs across different layers have a consistent mean and variance for stabilization. Given an input matrix $X \in \mathbb{R}^{m \times d}$, we have:

$$\text{LayerNorm}(X)_{i,j} = \frac{\gamma_j (x_{i,j} - \mu_i)}{\sigma_i} + \beta_j, \tag{3}$$

where $i \in [m]$, $j \in [d]$, $\mu = \sum_{i=1}^d x_i / d$ and $\sigma = \sqrt{\sum_{i=1}^d (x_i - \mu)^2}$ are mean and standard deviation, and γ and β are affine transform parameters.

The primary privacy concern arises because clients have no direct control over how their data is handled once it is sent to the server. They must trust that the server processes the data honestly and follows the agreed terms of service. However, there exists a risk that the server could misuse the data, including unauthorized processing, storing the data indefinitely, or even selling it to third parties. Even if the server is trustworthy, there are inherent risks associated with centralized data storage, such as data breaches or unauthorized access by malicious insiders. These risks are particularly concerning when dealing with sensitive or personally identifiable information. Therefore, while MLaaS offers significant convenience and computational power, it also necessitates careful consideration of privacy issues.

2.3 Privacy Preservation Techniques

This subsection introduces the popular cryptographic techniques involved in current PTI studies.

2.3.1 Functional Encryption. Functional encryption is a form of encryption that allows evaluating certain functions over encrypted data. The results of these functions are "leaked" from the ciphertexts, i.e., the result of the function is in plain data. This is beneficial for inference computation since only the first layer of the model needs to be run on encrypted data, and the rest can be run on plain data. However, this does leak some client data information to the server; the server learns the model output and the intermediate results of the computation.

2.3.2 Secure Multi-Party Computation (MPC). Secure MPC [14] refers to a collection of cryptographic algorithms and protocols, e.g., garbled circuits, secret sharing, and oblivious transfer, which allows multiple parties to jointly implement a computation task while keeping their respective data private. In essence, although all parties collaborate to perform a computation, no one can access information that compromises the privacy of others. The objective of MPC is to develop a secure protocol enabling these participants to collectively evaluate a function on their private inputs, ensuring that the output is accurate while safeguarding their private data, even in the presence of curious or malicious participants. Secure MPC can be formally described as follows: Consider n parties, denoted as $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$, where each party $\mathcal{P}_i$

holds a private input x_i. The goal is to jointly compute a predefined function $f(x_1, \ldots, x_n) \rightarrow y$, where y is the output obtained from the inputs of all parties. The computation results in $y = (y_1, \ldots, y_n)$, where each party $\mathcal{P}_i$ only learns y (or a portion of y) without gaining any knowledge of the other parties' inputs.

Upon these primitives, researchers have developed different protocols to implement model inference while preserving the privacy of the inputs. However, these protocols often require a lot of communication during the computation, which can make inference latency a problem.

Furthermore, we introduce some of the basic building blocks in MPC. It is worth noting that we mainly present MPC techniques involved in the PTI studies, and the reader is advised to read [14, 66] if interested in more MPC techniques.

- **Secrete Sharing (SS):** Secret sharing is a fundamental component of many MPC techniques. Typically, a (t, n)-secrete sharing scheme divides a secret S into n shares so that any $t - 1$ of the shares reveal no useful information about S. The secret can only be reconstructed from any combination of at least t shares.
- **Oblivious Transfer (OT):** The oblivious transfer allows a sender to transmit one of many pieces of information to a receiver without knowing which piece was received. A standard 1-out-of-k OT is denoted by k-OT, where a sender $\mathcal{S}$ has k messages $\{m_0, \ldots, m_{k-1}\}$ and a receiver $\mathcal{R}$ holds a choice bit $b \in [k]$. OT allows $\mathcal{R}$ to obtain m_b while learning nothing about other messages, and $\mathcal{S}$ learns nothing.

2.3.3 Homomorphic Encryption (HE). HE is a form of encryption that enables computations on encrypted data directly without decryption. The computations remain encrypted and, once decrypted, yield results identical to those obtained from the same operations on unencrypted data. Most HE schemes are based on public key cryptography, where a public key (pk) is used to encrypt data and a secret key (sk) to decrypt results. The public key can be shared freely for encryption purposes, whereas the secret key is required to decrypt messages. These schemes are secure because access to the public key does not compromise the private key. An overview of the main homomorphic operations is as below.

- $\mathrm{Enc}(\mathrm{pk}, m) \rightarrow c$: On public key pk and a plaintext m, perform encryption to obtain a ciphertext c.
- $\mathrm{Dec}(\mathrm{sk}, c) \rightarrow m$: On secret key sk and a ciphertext c, perform encryption to obtain the plaintext message m.
- $\mathrm{Eval}(\mathrm{pk}, c_0, \ldots, c_{k-1}, \mathrm{C}) \rightarrow \mathrm{Enc}(\mathrm{pk}, \mathrm{C}(m_0, \ldots, m_{k-1}))$: On public key pk, ciphertexts $c_1, \ldots, c_k$ encrypted from $m_1, \ldots, m_k$, and a circuit C (sequence of operations), outputs an encrypted computation result as same as the evaluation result of $\mathrm{Enc}(\mathrm{pk}, \mathrm{C}(m_0, \ldots, m_{k-1}))$.

Nonetheless, every operation on encrypted data introduces a small amount of noise, which accumulates as more operations are performed. Beyond a certain threshold, this noise can become large enough to prevent correct decryption. We can categorize HE schemes by the operations used in circuit C and its computational depth, i.e., the number of consecutive operations required for evaluation.

- **Partially Homomorphic Encryption:** Schemes support the evaluation of circuits consisting of a single type of operation—either addition or multiplication—but not both.
- **Somewhat Homomorphic Encryption (SHE):** SHE schemes enable a limited number of addition and multiplication operations on encrypted data, but this capability is restricted to a subset of computational circuits. The primary limitation arises from the accumulation of noise with each operation, which inherently restricts the depth of circuits that can be processed.
- **Fully Homomorphic Encryption (FHE):** FHE schemes allow for arbitrary operations on encrypted data, unrestricted by the type or the number of operations. A key feature of FHE is *bootstrapping*, a process designed to manage noise accumulation during operations. Bootstrapping involves decrypting a ciphertext c and then

encrypting it again, leading to a fresh ciphertext c' with a reset noise level. This could enable continuous computations on encrypted data. However, the computational intensity of bootstrapping makes FHE schemes still prohibitively expensive for many applications.

- **Leveled FHE.** Leveled FHE is designed to handle a predefined maximum number of operations on encrypted data. It supports both addition and multiplication, but requires the maximum circuit depth (number of operations) to be set in advance. Unlike standard FHE, Leveled FHE does not require the computationally expensive bootstrapping process. This allows it to provide an exact and adaptable depth of computation that can be precisely tailored for specific tasks, offering a balance between computational power and efficiency.

3 PIVACY THREATS IN SECURE INFERENCE

Semi-honest. The semi-honest security model is based on the assumption that all parties involved in computation will honestly follow the established protocols while passively attempting to gather extra private information during its execution. This model is often used when the parties have a basic level of trust in each other, not to actively disrupt the process. Semi-honest security is a common assumption for privacy-preserving machine learning (PPML). The parties have to trust each other to some extent so that they can jointly contribute to the result of inference.

Honest-Majority. The honest-majority security model operates under the assumption that as long as the majority of the participants are honest and follow the protocol, they can prevent a minority who may be dishonest from tampering with the process or stealing private information. It is suited for situations involving more participants, where not everyone might be fully trustworthy. This model enhances security by allowing the computation to tolerate some level of misbehavior from a minority of the participants without compromising the integrity or confidentiality of the overall process.

4 AN OVERVIEW OF PRIVATE TRANSFORMER INFERENCE STUDIES: SETUPS, CRYPTOGRAPHIC APPROACHES, PROS AND CONS

This section provides an overview of state-of-the-art private transformer inference (PTI) studies, along with a critical review of their strengths and weaknesses. Specifically, we first introduce a secure inference system setup in Section ??. Then, we review current PTI studies and their privacy-preserving approaches in Section 4.2. A discussion about strengths and weaknesses is finally presented in Section 4.3.

4.1 Bottlenecks in private transformer inference

In recent years, there has been significant development in neural network private inference for traditional CNN (Convolution Neural Network) and RNN (Recurrent Neural Network) models [27, 51]. However, due to essentially different structures, private transformer inference brings several new challenges.

Large Matrix Multiplications. First, transformer models involve a huge number of large matrix multiplications rather than matrix-vector multiplications widely studied in CNN. In addition, a transformer model usually consists of multiple encoders (i.e., decoders), and the multiplication scale is much higher than the traditional networks. Direct extensions of existing matrix-vector multiplication protocols to transformers typically result in unaffordable overheads [8, 18].

Complex Nonlinear Functions. Traditional neural networks commonly employ crypto-friendly non-linear functions, e.g., Rectified Linear Unit (ReLU), Maxpool and batch normalization. In Contrast, transformers extensively employ Gaussian Error Linear Unit (GeLU), Softmax and LayerNorm functions. These complex nonlinear functions are critical

to the inference performance but are not computation-friendly for encryption primitives, thus making inference less efficient.

4.2 An overview of PTI studies with employed cryptographies

We examine the cutting-edge PTI studies (2022-2024) in Table 4, including their setups, employed tools, threat models and improved components. In particular, the studies are classified and discussed based on their running setups (2PC, 2PC-Dealer and 3PC). We believe that the choice of setups significantly affects the tools employed, the threat model, and the components to be improved. For instance, secure matrix multiplication requires additional privacy protection (e.g., HE) and overheads under the 2PC setup and is therefore optimized in all 2PC-based studies, but not in 2PC-Dealer and 3PC setups [1, 8, 12, 17, 28, 32, 36, 50, 59].

Studies based on 2PC [7, 11, 18, 20, 34, 43, 64] follow the most common server-client-participated computation paradigm and often assumes a standard semi-honest (i.e., honest-but-curious) threat model. The two parties, often server and client, will honestly follow the inference protocol while passively attempting to gather extra private information during its execution, which is a natural and practical consideration for current MLaaS platforms. In particular, THE-X [7] and NEXUS [64] manage to implement purely HE-based PTI frameworks, which requires them to optimize almost all nonlinear layers, as nonlinearity is often not supported by HE. In contrast, other studies [11, 18, 20, 34, 43] favor MPC+HE hybrid frameworks to improve the performance (e.g., inference speed, model accuracy) at the cost of additional complexity in the setups. As we have discussed, secure matrix multiplication in 2PC is the bottleneck for optimization as it requires additional privacy protection.

In both 2PC-Dealer and 3PC setups, the introduction of a "helper party" has effectively eliminated matrix multiplication as a significant bottleneck. However, nonlinear layers, especially Softmax and GeLU, now represent the primary sources of overhead in MPC [12, 28, 59]. GeLU requires a high-order Taylor expansion involving many multiplications, while Softmax demands iterative squaring for exponentials and comparisons for numerical stability. Hence, we can see from Table 4 that most studies manage to improve the performance of Softmax and GeLU. Apart from standard semi-honest assumption [8, 12, 17, 28, 36, 50, 59], PrivFormer [1] and PPTIF [32] can work under the honest-majority setup, where two out of three parties are honest adversaries, and the last one can be semi-honest or even malicious. Nonetheless, even though 2PC-Dealer and 3PC setups could improve secure inference performance efficiently, the assumption of a trusted dealer or non-colluding server parties is sometimes considered unrealistic in practice [20].

4.3 Strengths and Weaknesses.

MPC and HE both enable secure computation, and there is no clear better technique. Instead, the choice has to be made based on multiple factors depending on the applications. Here, we provide a critical review of their strengths and weaknesses in terms of communication cost, computational overhead and privacy protection.

Client Computation. Some studies require the client to frequently participate in the computation during the inference process, and hence they tend to assume that clients also have relatively high computing resources. However, involving clients in computation could pose significant consumption challenges to low-power devices in practical, e.g., mobile devices. THE-X [7]

Model Performance. THE-X [7] and MPCFormer [28] simply replace the nonlinear layers with cryptography-friendly approximations for efficiency and, therefore, suffer a significant model performance degradation. To cure the accuracy drop, MPCFormer [28] applies a knowledge distillation (KD) [19] method to train the approximated model,

Table 1. An overview of secure transformer inference studies with employed cryptographies.

Setup	Study	Tools	Improved Components				Experiments on
			MatMul	GeLU	Softmax	LayerNorm	
2PC	THE-X [7]	HE	○	●	●	●	BERT-Tiny
	Iron [18]	ASS+HE	●	●	●	○	BERT-Base
	CipherGPT [20]	ASS+HE	●	●	○	○	GPT2-Base
	Bumblebee [34]	ASS+HE	●	●	●	○	BERT-Base/Large, GPT2-Base, LLaMA-7B, ViT-Base
	East [11]	ASS+HE	●	●	●	●	BERT-Base
	Zimerman et al. [69]	HE	●	●	●	●	ViT, Swin Transformer
	BOLT [43]	ASS+HE	●	●	●	○	BERT-Base
	NEXUS [64]	HE	●	●	●	●	BERT-Base, GPT-2
	CipherFormer [58]	HE+GC	●	●	●	●	Transformer
	MPCViT [63]	ASS	●	●	●	○	CCT, CVT
	SAL-ViT [65]	ASS	●	○	●	○	CCT
	Primer [67]	HE+GC	●	○	●	○	BERT-Tiny/Small/Base/Medium/Large
	PrivCirNet [60]	MPC+HE	●	○	○	○	ViT
	Liu et al. [30]	MPC+HE	●	●	●	●	BERT-Tiny/Medium, Roberta-Base
	RNA-ViT [6]	MPC	●	●	●	○	CCT
	Powerformer [44]	HE	●	●	●	●	BERT-Tiny
	Rovida et al. [49]	HE	●	●	●	●	BERT-Tiny
	Zimerman et al. [68]	HE	●	●	●	○	Roberta, GPT, ViT
	SecureGPT [62]	MPC	●	●	●	○	GPT
	SecBERT [21]	MPC	●	●	●	○	BERT
	THOR [41]	HE	●	●	●	○	BERT-Base
2PC-Dealer	MPCFormer [28]	MPC	○	●	●	○	BERT-Base
	Wang et al. [59]	ASS	○	○	●	○	XLM, ViT
	SIGMA [17]	FSS	○	●	●	●	BERT-Tiny/Base/Large, GPT2/GPT-Neo, Llama2-7B/13B
	SecFormer [36]	ASS	○	●	●	●	BERT-Base/Large
	SecureTLM [8]	HSS	●	●	●	●	-
	Curl [50]	LUT	○	●	●	○	BERT-Tiny/Base/Large GPT2/GPT-Neo
3PC	PUMA [12]	RSS	○	●	●	○	BERT-Base/Large, GPT2-Base/Medium/Large, Roberta-Base, LLaMA-7B
	PrivFormer [1]	RSS	○	○	●	○	-
	PPTIF [32]	RSS	●	○	●	○	-

using teacher guidance from the original transformer. However, the additional distillation period also brings extra computation overheads to the framework.

Table 2. Strengths and weaknesses to current secure transformer inference studies.

Study	Overhead	Client Computation	Model Performance	Hardware Acceleration	Client Data Privacy	Model Privacy	Scalability	Code Availability	Comments
THE-X [7]	-		-		+	-	-	-	Lack of strict privacy protection for intermediate inference results
MPCFormer [28]	+	-			+	+	-	+	Require a trusted dealer and additional distillation training
Wang *et al.* [59]	+	-	?		+	+		-	Require a trusted dealer
Iron [18]	-	-	+		+	+		-	
CipherGPT [20]		-	+		+	+		-	
Bumblebee [34]		-	+		+	+	+	-	
SIGMA [17]	+	-	+	+	+	+	+	-	Require a trusted dealer
PUMA [12]	+	-	+		+	+		+	Operate under 3PC setup
PrivFormer [1]	-	+	?		+			-	Operate under 3PC setup
PPTIF [32]	-	+	?		+	+		-	Operate under 3PC setup
BOLT [43]		-	+		+	+		-	
NEXUS [64]	-	+	+		+	+		-	Non-interactive protocol for inference

Table 3. Links to available implementations.

Study	Year	Link	Comment
BumbleBee [34]	2023	https://github.com/AntCPLab/OpenBumbleBee	-
BOLT [26]	2024	https://github.com/Clive2312/BOLT	-
NEXUS [64]	2024	https://github.com/zju-abclab/NEXUS	-
Zama-AI	-	https://github.com/zama-ai/concrete-ml/tree/release/1.1.x/use_case_examples/llm	-

Later studies [12, 17, 18, 20, 34, 43, 64] have paid more attention to model inference performance. For the nonlinear layers, they often utilize look-up tables or high-order polynomial approximations to maintain accuracy while ensuring cryptographic compatibility. Hence, they can obtain satisfying performance on GLUE benchmarks [57], comparable to the original transformer. Stuides [1, 32, 59] focus more on the general inference framework of transformer encoder (decoder) and does not report performance results on specific intelligence tasks.

Code Availability. We provide links to the implementations of those open-source PTI research in Table 3.

5 LINEAR LAYERS IN TRANSFORMERS

Linear layers in transformers mainly involve large matrix multiplications (MatMul) in attention and feed-forward layers, and this section discusses the state-of-the-art literature on secure MatMul in PTI. Specifically, Section **??** first provides a breakdown of linear layers in a standard Transformer encoder. Then, Section 5.1 generalizes the two kinds of main-stream secure MatMul protocols in current PTI studies.

Table 4. An overview of PTI studies with employed MPC backbones.

#Parties	Study	SS Scheme	MPC Engine	HE Method	HE Library
2PC	THE-X [7]	-	-	-	SEAL
	Iron [18]	2-out-of-2 ASS	EzPC	BFV	SEAL
	CipherGPT [20]	2-out-of-2 ASS	-	BFV	SEAL
	Bumblebee [34]	2-out-of-2 ASS	-	BFV	SEAL
	East [11]	2-out-of-2 ASS	EzPC	BFV	SEAL
	BOLT [43]	2-out-of-2 ASS	EzPC	BFV	SEAL
	NEXUS [64]	-	-	RNS-CKKS	SEAL
	MPCViT [63]	2-out-of-2 ASS	SecretFlow-SPU	-	-
	SAL-ViT [65]	2-out-of-2 ASS	CrypTen	-	-
	SecFormer [36]	2-out-of-2 ASS	CrypTen	-	-
	Primer [67]	2-out-of-2 ASS	-	-	-
	PrivCirNet [60]	-	-	BFV	SEAL
	RNA-ViT [6]	2-out-of-2 ASS	CrypTen	-	-
	Liu *et al.* [30]	2-out-of-2 ASS	OpenCheetah, SCI	BFV	-
	PowerFormer [44]	-	-	RNS-CKKS	Lattigo
	Rovida *et al.* [49]	-	-	RNS-CKKS	OpenFHE
	Zimerman *et al.* [68]	-	-	CKKS	HEaaN
	SecureGPT [62]	-	emp-toolkit	-	-
	SecBERT [21]	2-out-of-2 ASS	ABY	-	-
	THOR [41]	-	-	RNS-CKKS	Liberate.FHE
2PC-Dealer	MPCFormer [28]	2-out-of-2 ASS	Crypten	-	-
	Wang *et al.* [59]	2-out-of-2 ASS	Crypten	-	-
	SIGMA [17]	FSS	Orca	-	-
	SecFormer [36]	2-out-of-2 ASS	Crypten	-	-
	SecureTLM [8]	HSS	EzPC	-	-
	Curl [17]	LUT	-	-	-
3PC	PUMA [12]	2-out-of-3 RSS	SecretFlow-SPU	-	-
	PrivFormer [1]	2-out-of-3 RSS	Falcon	-	-
	PPTIF [32]	2-out-of-3 RSS	MP-SPDZ	-	-

Table 5. Links to available backbones.

Study	Year	Link	Comment
EzPC [5]	2019	https://github.com/mpc-msri/EzPC	-
MP-SPDZ [25]	2020	https://github.com/data61/MP-SPDZ	-
Falcon [56]	2021	https://github.com/snwagh/falcon-public	3PC
CrypTen [26]	2021	https://github.com/facebookresearch/CrypTen	-
SecretFlow-SPU [37]	2023	https://github.com/secretflow/spu	-

Based on the above analysis, we can see that high-dimensional **matrix multiplication (MatMul)** is the primary linear computation involved in transformers. Compared to the matrix-vector multiplications used in traditional networks (e.g., CNNs), MatMuls require significantly more computational resources due to the increased dimensionality and

complexity [18, 43]. This is particularly evident in the attention mechanism: (??)-(??). Thereafter, how to design efficient and secure MatMul protocols is crucial for PTI studies.

5.1 Large Matrix Multiplications

The inference of a transformer-based model can involve hundreds of large matrix multiplications. For ease of understanding, we first present an example of MatMul operation as follows:

Example 5.1. Given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times k}$, the matrix multiplication output C is:

$$C = A \times B, \tag{4}$$

where $C \in \mathbb{R}^{m \times k}$, and its each element $C_{i,j}$ can be regarded as:

$$C_{i,j} = \sum_{k=1}^{N} A_{i,k} \cdot B_{k,j}, \tag{5}$$

where $A_{i,k}$ is an element from the i-th row of A and $B_{k,j}$ is an element from the j-th column of B.

According to (5), we can see that each element of the MatMul output C can be computed as the sum of products of corresponding elements from the rows of A and the columns of B. This naive finding allows us to break the whole matrix multiplication into a series of addition and multiplication operations, and cryptographic primitives can be adapted to support these operations.

Particularly, to implement secure MatMul protocols, current PTI studies adopt the following main-stream ways:

- **Secret Sharing (SS)-based Scheme**: In the SS-based scheme, the matrices A and B are divided into shares and distributed among multiple parties. Each party holds a share of A and B but not the original matrices. The matrix multiplication is performed on the shares, and the results are combined to obtain the final result.
  Strengths:
  – Additions can be easily evaluated locally "for free", dramatically saving communication resources.
  Weaknesses:
  – Multiplications need extra communication between parties.
- **Homomorphic Encryption (HE)-based Scheme**: Here, one matrix (A or B) is encrypted using HE that supports arithmetic operations on ciphertexts. The two matrices are then multiplied, and the MatMul output is decrypted to obtain the final product.
  Strengths:
  – HE-based methods naturally support mixed computations, which is communication-friendly.
  Weaknesses:
  – Operations involving ciphertexts require more computation and are applied component-wisely by default, resulting in a considerable computational overhead.

Next, we further detail state-of-the-art PTI studies on SS-based MatMul protocols and HE-based MatMul protocols in Section 5.2 and Section 5.3, respectively.

5.2 Secrete Sharing-based MatMul protocols

We first introduce secrete sharing-based MatMul protocols in current secure transformer inference studies. Specifically, secret sharing schemes support adding secret-shared values locally (without interaction) by simply adding the

corresponding secret shares, making additions relatively inexpensive. Hence, the key point is efficiently implementing multiplications while preserving privacy. Various studies have explored different techniques for secure multiplication tailored to the specific MPC settings and SS mechanisms they employ. Table 6 presents their utilized MPC settings, SS schemes and techniques for multiplications, along with a critical review of their strength and weaknesses in overheads.

Iron [18] builds its MatMul protocol on the 2PC setting and 2-out-of-2 ASS scheme. In this setup, direct multiplication would cause both parties to interact in a way that could reveal information about their shares. Iron leverages the Brakerski-Fan-Vercauteren (BFV) HE scheme to encrypt one party's share before communication, allowing the multiplication to be performed on encrypted data (ciphertexts). To manage the computational complexity, Iron packs the plaintext matrices into polynomials before applying HE encryption. This packing strategy is based on the insight that, by arranging the polynomial coefficients in a specific manner, the resulting polynomial multiplication corresponds directly to the desired MatMul results [22]. Similar to Iron, BumbleBee [34] also combines ASS with HE and starts with the coefficient encoding approach. However, it further improves the packing scheme with a procedure named *IntrLeave*, which could efficiently interleave multiple polynomials into one, reducing the size of ciphertexts for transmission. Consequently, it requires much less communication compared to Iron. CipherGPT [20] accelerates MatMuls by leveraging a preprocessing phase with sVOLE [4] correlations and batching multiple matrix multiplications together, reducing both computation and communication consumptions.

MPCFormer [28] and Wang *et al.* [59] operate under the 2-out-of-2 ASS but introduces an additional trusted third-party dealer. The role of the dealer is to provide a secret shared triple, $c = ab$, to two parties and facilitate the multiplication operation, and such a joint computation technique is called Beaver triple [3]. The introduction of Beaver triple requires one extra round of communication for multiplication compared to direct multiplication without MPC. However, neither MPCFormer nor Wang *et al.* [59] uses other additional techniques to accelerate matrix multiplication, and they both claim that the nonlinear layers in Transformers are the major sources of bottlenecks. In particular, matrix multiplications account for only 12.7% of the overall BERT-Base model runtime under MPC, and the communication overhead is also much lower than that of the non-linear layer. Privformer [1] and PUMA [12] build their MatMul protocols upon the setup of 3PC and 2-out-of-3 RSS scheme. All three parties are directly involved in the MPC computations. For multiplication, each of the three parties computes part of the result locally, then exchanges noisy results via the *re-sharing* method in Araki *et al.* [2], and finally combines them into a new shared product value.

Generally, SS-based MatMul approaches above mainly rely on either 2PC+HE [18, 20, 34] or introduce an additional helper party to participate in computations [1, 12, 28, 59]. The former is more convenient for practical application in secure inference, as the server and the client can act as two computational parties, eliminating the need for additional assumptions about parties. However, 2PC still poses a very large overhead due to using additional cryptography to support multiplication [47]. In contrast, 2PC-Dealer and 3PC setups are effective in improving the efficiency of MatMuls, but they rely on a trusted dealer or honest-majority assumption. It is difficult to compare the two approaches in a strict sense because of their disparities in the security assumption and applicable scenarios.

5.3 Homomorphic Encryption-based MatMul protocols

The use of HE for matrix multiplications in transformers is still relatively unexplored. There is little work on PTI using HE-based protocols for matrix multiplications. While matrix multiplication is inherently compatible with HE, as it involves only additions and multiplications, a straightforward implementation can be highly inefficient. Specifically, HE-based MatMul protocols in transformers still face several key challenges:

Table 6. A Comparison of SS-based MatMul Protocols.

Study	MPC Setup	SS Scheme	Tech. for Mul.[1]	Comm.[2]	Comp.[3]	Comments
Iron [18]	2PC	2-out-of-2 ASS	HE	-		
BumbleBee [34]	2PC	2-out-of-2 ASS	HE			
CipherGPT [20]	2PC	2-out-of-2 ASS	HE		+	
MPCFormer [28]	2PC-Dealer	2-out-of-2 ASS	Beaver triple			Needs a trusted dealer
Wang *et al.* [59]	2PC-Dealer	2-out-of-2 ASS	Beaver triple			Needs a trusted dealer
PUMA [12]	3PC	2-out-of-3 RSS	re-sharing	+	+	Honest-majority
Privformer [1]	3PC	2-out-of-3 RSS	re-sharing	+	+	Honest-majority

[1] Techniques for Multiplications.   [2] Communication.   [3] Computation.

Note that + indicates a strength and - to a weakness. A blank field means a solution is neither strong nor weak in this area.

- **Transmission Overheads:** Ciphertext matrices are considerably larger than plaintext matrices, leading to substantial transmission costs.
- **Computational Expense:** A vast number of multiplications are required, and these multiplications are computationally expensive in the context of HE.
- **Limited Multiplicative Depth:** The multiplicative depth in HE is constrained by the accumulation of noise, often necessitating expensive bootstrapping operations to refresh the depth and enable further computations.

These deficiencies motivate the need for more efficient HE-based MatMul protocols, and we report the comparison of HE-based MatMul protocols in Table 7.

An early study, THE-X [7], first proposes to utilize an FHE scheme, CKKS, to implement secure transformer inference. It naively converts all matrix multiplications into the element-wise style and employs no other acceleration techniques. Hence, it only evaluates a small transformer model BERT-Tiny and does not report running time performance.

BOLT [43] supports MatMuls in transformers using a leveled FHE scheme, BFV. It employs a column-wise packing for the ciphertext matrix and adopts Gazelle's diagonal packing [24] for the plaintext matrix. Ciphertext packing is to organize and pack multiple plaintext data into a single ciphertext, which allows for parallel operations using SIMD techniques. BOLT also utilizes the outer product representation and partial rotation techniques to speed up the ciphertext-ciphertext MatMuls (e.g., $Q \cdot K^T$ in the attention layer). BOLT operates under an MPC-HE hybrid framework, which allows it to automatically reset the HE noise without bootstrapping operations.

The latest study, NEXUS [64], operates with RNS-CKKS and incorporates SIMD ciphertext compression. This technique compresses multiple ciphertexts into a single one, thereby reducing the volume of data transferred. Decompression is achieved with a fixed number of operations, adding no additional computational overhead. Additionally, NEXUS employs SIMD slots folding, accelerating ciphertext-ciphertext MatMuls. This is accomplished by efficiently aggregating element-wise products within a ciphertext using significantly fewer rotations, thus enhancing processing speed.

Compared to secrete-sharing based MatMuls, HE often allows for more savings on communication bandwidth and roundtrips required for computation. However, computations on ciphertexts are computationally expensive, and large MatMuls in transformers could further increase multiplicative depth, complicating ciphertext-ciphertext matrix multiplication. Current studies [43, 64] mainly leverage ciphertext packing schemes and SIMD techniques to address the above challenges. This approach significantly reduces the number of ciphertexts and speeds up computations by enabling efficient parallel processes.

Table 7. A Comparison of HE-based MatMul Protocols.

Study	HE Scheme	Packing	SIMD	Bootstrapping	Comm.[1]	Comp.[2]	Comments
THE-X [7]	CKKS	$\times$	$\times$	?	-	-	Tiny evaluation model
BOLT [43]	BFV	$\checkmark$	$\checkmark$	$\times$	+	+	
NEXUS [64]	RNS-CKKS	$\checkmark$	$\checkmark$	$\checkmark$	+		

[1] Communication.   [2] Computation.

Note that + indicates a strength and - to a weakness. A blank field means a solution is neither strong nor weak in this area. And ? means the information is missing in the study.

6 NON-LINEAR LAYERS

Securing the non-linear functions in Transformers poses another challenge due to their complexity in cryptographic primitives. The bulk of the total run time is from non-linear layers for both MPC and HE studies [18, 28, 43]. Non-linear functions in Transformers include Softmax, GeLU and LayerNorm. We detail each of them in the following subsections.

6.1 Softmax

In the attention mechanism, one key challenge is the implementation of the nonlinear Softmax operation. For the sake of numerical stability [16], when given an input vector $x \in \mathbb{R}^d$, the Softmax function can be computed as:

$$\text{Softmax}(x) = \left[\frac{\exp(x_i - \hat{x})}{\sum_{j=1}^d \exp(x_j - \hat{x})} \bigg|_{i \in [d]} \right], \tag{6}$$

where $\hat{x} = \max_{i \in [d]} x_i$. It is worth noting that all inputs to the exponential function in (6) are non-positive now. The main challenge is to efficiently calculate the underlying exponential function and division computation. Next, we discuss recent research advancements in optimizing Softmax and provide a detailed comparison in Table 8.

Tech Tips: In essence, the Softmax function could be regarded to perform a re-weight normalization of the obtained attention map, and its general form is formulated as follows:

$$\text{Softmax}(x_i) = \frac{F(x_i)}{\sum_{j=1}^d F(x_j) + \epsilon}, \tag{7}$$

where function $F(\cdot)$ widely adopts the exponential function $\exp(\cdot)$ and ϵ is a small positive number to avoid a zero denominator. Therefore, existing studies mainly optimize its computation in the following two methods:

- Employ more crypto-friendly functions to server as $F(x)$ in Softmax, such as the $\text{ReLU}(x)$ and quadratic function $(x + c)^2$. Those functions often consist of more basic arithmetic operations (e.g., multiplications and comparisons), the overhead to compute $\text{Softmax}(x_i)$, thus dramatically decreasing overheads. It is worth noting that this approach comes at the cost of a significant drop in accuracy.
- Directly utilize different polynomial approximations (e.g., the Taylor series) of $\exp(x)$ to serve as $F(x)$. This method could help to maintain the accuracy. However, one should be careful about the choice of polynomial approximation order; too high a polynomial order can bring better accuracy while reducing computational efficiency obviously.

Next, we classify the selected PTI studies based on the Tech Tips above and discuss how they cope with the Softmax function in more detailedly.

Crypto-friendly Functions. Studies [6, 28, 36, 63, 65] that favour the first solution often use aggressive crypto-friendly functions and, therefore, require additional training to minimize the loss of accuracy. Specifically, we present their methods as follows:

$$\text{Softmax}(x_i) \approx \frac{F(x_i)}{\sum_{j=1}^{d} F(x_j) + \epsilon}, \text{ for } F(x) = \begin{cases} (x+c)^2, & \text{in MPCFormer [28], SecFormer [36]} \\ \text{ReLU}(x), & \text{in MPCViT [63]} \\ (a \cdot x + c)^2, & \text{in SAL-ViT [65]} \\ (x+c)^4, & \text{in RNA-ViT [6]} \end{cases} \quad (8)$$

It is obvious that (8) and the original Softmax (6) differ a lot by numerical values. Hence, the above studies all utilize the Knowledge Distillation (KD) [19] method to bridge the performance gap.

Polynomial Approximations. Studies [12, 20, 34, 43, 64] employ polynomial approximation to replace the original $\exp(x)$ in Softmax. The computation of polynomials can be decomposed into basic addition and multiplication operations, which are easily supported by cryptographic techniques. BOLT [43] decomposes the input $x_i - \hat{x}$ into a non-negative integer z and a float number $p \in (-\ln 2, 0]$, where $x_i - \hat{x} = (-\ln 2) \cdot z + p$. Then, the exponentiation is approximated as $\exp(\widetilde{x}_i) = \exp(p) \gg z$, where $\gg$ denotes the right shift operator. Since the range of p is relatively small, a 2-degree polynomial is enough for the approximation:

$$\exp(p) \approx 0.3585(p + 1.353)^2 + 0.344. \quad (9)$$

For the $\frac{1}{\sum_{j \in [d]} \exp(x_j - \hat{x})}$ part, i.e., the reciprocal of the summed exponential function's outputs, BOLT uses the reciprocal protocol from SIRNN [48]. Besides, NEXUS [64], CipherGPT [20], PUMA [12] and BumbleBee [34] all approximate the exponential function in Softmax using the Taylor series.

$$\exp(x) \approx \begin{cases} 0, & \text{if } x \leq a \\ (1 + \frac{x}{2^r})^{2^r}, & \text{if } a \leq x \leq 0 \end{cases}, \text{ for } \begin{cases} a = -\inf, r = 6, & \text{in NEXUS [64]} \\ a = -13, r = 6, & \text{in BumbleBee [34]} \\ a = -16, & \text{in CipherGPT [20]} \\ a = -14, r = 5, & \text{in PUMA [12]} \end{cases} \quad (10)$$

Additionally, NEXUS uses its proposed QuickSum and the Goldschmidt division algorithm to compute the reciprocal of the sum of exponential functions. Rovida *et al.* [49] approximates Softmax using the Maclaurin series:

$$\exp(x) \approx \sum_{i=0}^{6} \frac{x^i}{i!}, \quad (11)$$

and for the division operation, they use a Chebyshev polynomial approximation:

$$\frac{1}{x} \approx \frac{c_0}{2} + \sum_{i=1}^{119} c_i T_i(x). \quad (12)$$

LUTs. Some studies [17, 18] use look-up tables (LUTs) for Softmax. A LUT is a pre-computed table stored in memory, containing input-output pairs. It allows for quick retrieval of function values by directly looking up results, eliminating the need for real-time calculation. Iron [18] uses a tree-reduction protocol to evaluate the max function and then

invokes the LUT-based exponential and reciprocal protocols [48] to evaluate Softmax. Sigma [17] optimizes Softmax by introducing efficient methods for calculating the max, negative exponential, and inverse functions using reduced bit-width LUTs, significantly improving computational efficiency and reducing resource consumption. Generally speaking, LUTs can significantly reduce the computation time but typically incur a higher communication overhead, which can be validated in Table 8 (e.g., Sigma [17] to BumbleBee [34], Iron [18] to BOLT [43]).

Table 8. A Comparison of Softmax Approximations.

Study	Input	Comm. (MB)	Comm. set.	Runtime (s)	Error	Comments
MPCFormer [28]	-	-	-	4.9	-	Require distillation
THE-X [7]	-	-	-	-	-	Require training
PUMA [12]	-	-	-	-	-	No breakdown provided
BumbleBee [34]	$\mathbb{R}^{128\times128}$	162.24	LAN[1]/WAN[2]	2.11/5.79	$< 2^{-10}$	-
CipherGPT [20]	$\mathbb{R}^{256\times256}$	277.59	LAN[3]	14.865	-	-
Sigma [17]	$\mathbb{R}^{128\times128}$	266.24	LAN[4]	0.44	-	-
Iron [18]	$\mathbb{R}^{128\times128}$	3596.32	LAN[3]/WAN[5]	60/1900	3.2×10^{-5}	-
BOLT [43]	$\mathbb{R}^{128\times128}$	1447.65	LAN[3]/WAN[5]	16/775	1.4×10^{-6}	-
NEXUS [43]	$\mathbb{R}^{128\times128}$	0	LAN[3]/WAN[5]	242/242	3.1×10^{-6}	Nearly comm. free

[1]1Gbps, 0.5ms   [2]400Mbps,4ms   [3]3Gbps, 0.8ms   [4]9.4Gbps, 0.05ms   [5]100Mbps, 80ms

6.2   GeLU

Rather than crypto-friendly ReLU, Transformers utilize GeLU activations. The GeLU (Gaussian Error Linear Unit) activation function is defined as follows:

$$\text{GeLU}(x) = \frac{x}{2}\Big(1 + \text{erf}\big(\frac{x}{\sqrt{2}}\big)\Big), \tag{13}$$

where $\text{erf}(\cdot)$ denotes the Gaussian error function which is given by $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$. In many libraries, e.g., PyTorch, the following approximation is provided:

$$\text{GeLU}(x) \approx \frac{x}{2}\Big(1 + \tanh\big(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3)\big)\Big). \tag{14}$$

and where:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{15}$$

Hence, the main challenge is how to efficiently handle the nonlinear $\text{erf}(x)$ (i.e., $\tanh(x)$) part.

> **Tech Tips:** The function $\text{GeLU}(x)$. Besides, polynomials are still available to approximate either $\tanh(x)$ (i.e., $\text{erf}(x)$) or the entire $\text{GeLU}(x)$

Some studies [7, 28] approximate GeLU simply using crypto-friendly ReLU or ReLU approximation. For example, THE-X [7] and PowerFormer [44] directly replaces the GeLU layers with ReLU activation functions:

$$\text{GeLU}(x) \approx \text{ReLU}(x) = \max(0, x), \tag{16}$$

Table 9. A Comparison of GeLU Approximations.

Study	Input	Comm. (MB)	Comm. set.	Runtime (s)	Error	Comments
THE-X [7]	-	-	-	-	-	Require training
MPCFormer [28]	-	-	-	-	-	Require distillation
PUMA [12]	-	-	-	-	-	No breakdown provided
BumbleBee [34]	$\mathbb{R}^{128\times3072}$	162.24	LAN[1]/WAN[2]	2.11/5.79	$< 2^{-10}$	-
CipherGPT [20]	$\mathbb{R}^{256\times3072}$	575.91	LAN[3]	20.657	-	-
SIGMA [17]	$\mathbb{R}^{128\times3072}$	163.84	LAN[4]	0.25	-	
Iron [18]	$\mathbb{R}^{128\times3072}$	7960.00	LAN[3]/WAN[5]	126/4118	5.8×10^{-4}	
BOLT [43]	$\mathbb{R}^{128\times3072}$	1471.67	LAN[3]/WAN[5]	14/774	9.8×10^{-4}	
NEXUS [43]	$\mathbb{R}^{128\times3072}$	0	LAN[3]/WAN[5]	233/233	7.7×10^{-4}	Nearly comm. free

[1]1Gbps, 0.5ms   [2]400Mbps,4ms   [3]3Gbps, 0.8ms   [4]9.4Gbps, 0.05ms   [5]100Mbps, 80ms

where the computation of function $\max(\cdot)$ is left to the user to complete after decryption. Obviously, such an aggressive approximation is not accurate enough. MPCFormer [28] takes a quadratic function to approximate GeLU:

$$\text{GeLU}(x) \approx 0.125x^2 + 0.25x + 0.5. \tag{17}$$

However, Equa. (17) is designed for ReLU approximation in Chou et al. [9] and thus also introduces a significant accuracy drop (MPCFormer mitigates this drop by distillation).

$$\text{GeLU}(x) \approx \begin{cases} \max(0, x), & \text{in THE-X [7], PowerFormer [44]} \\ 0.125x^2 + 0.25x + 0.5, & \text{in MPCFormer [28]} \\ \max(0, x) + a * \min(0, x), & \text{in RNA-ViT [6]} \end{cases} \tag{18}$$

Some studies [17, 18, 20] utilize LUT-based methods for GeLU. Iron [18] notices that the sign of $\tanh(x)$ is equal to that of x. Hence, it first generates a negative input $\bar{x}$ where $|\bar{x}| = |x|$, and then the following two cases hold: $\tanh(x) = \tanh(\bar{x})$ for $x \leq 0$ and $\tanh(x) = -\tanh(\bar{x})$ for $x > 0$. Then, similar to dealing with Softmax, it calls the LUT-based protocols from SIRNN [48] to cope with the tanh function. SIGMA [17] finds the near-linearity and symmetry of GeLU, and it further employs a function $\delta(x)$ to measure the difference between GeLU(x) and ReLU(x) for $x \in [-4, 4]$:

$$\text{GeLU}(x) \approx \begin{cases} 0, & \text{if } x < -4 \\ \text{ReLU}(x) - \delta(x), & \text{if } -4 \leq x \leq 4 \\ x, & \text{if } x > 4 \end{cases} \tag{19}$$

where ReLU(x) is implemented by its proposed DReLU protocol, and $\sigma(x)$ is computed through the LUT.

Some studies [12, 34, 43, 64] employ polynomials to approximate GeLU efficiently. PUMA [12], BumbleBee [34] and NEXUS [64] notice that GeLU function is almost linear on the two sides, i.e., GeLU$(x) \approx 0$ for $x < -4$ and GeLU$(x) \approx x$

for $x > 3$. They also suggest an efficient low-degree polynomial for approximation within the short interval around 0:

$$\text{GeLU}(x) \approx \begin{cases} -c, & \text{if } x < a \\ \sum_{i=0}^{3} a_i x^i, & \text{if } a < x \le b \\ \sum_{i=0}^{6} b_i x^i, & \text{if } b < x \le 3 \\ x - c, & \text{if } x > 3 \end{cases}, \text{ for } \begin{cases} a = -4, b = -1.95, c = 0, & \text{in PUMA [12], NEXUS [64]} \\ a = -5, b = -2, c = 10^{-5}, & \text{in BumbleBee [34]} \end{cases} \tag{20}$$

Similarly, BOLT [43] presents an approximation using a 4-degree polynomial:

$$\text{GeLU}(x) \approx \begin{cases} x, & \text{if } x > 2.7 \\ \sum_{i=0}^{4} a_i |x|^i + 0.5x, & \text{if } |x| \le 2.7 \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

SecFormer [36] introduces the Fourier series to approximate the $\text{erf}(x)$:

$$\text{erf}(x) \approx \begin{cases} -1, & \text{if } x < -1.7 \\ \boldsymbol{\beta} \odot \sin{(\boldsymbol{k} \odot \pi x / 10)}, & \text{if } -1.7 \le x \le 1.7 \\ 1, & \text{otherwise} \end{cases} \tag{22}$$

6.3 Layer Normalization

Layer normalization ensures that the inputs across different layers of the network have a consistent mean and variance, helping to stabilize deep neural networks. It is applied after each attention block and feed-forward layer. For a given vector $\boldsymbol{x} \in \mathbb{R}^d$, the Layer normalization function is defined as follows:

$$\text{LayerNorm}(\boldsymbol{x}) = \left[\frac{\gamma(x_i - \mu)}{\sigma} + \beta \mid_{i=1,\dots,d} \right], \tag{23}$$

where $\mu = \sum_{i=1}^{d} x_i / d$ and $\sigma = \sqrt{\sum_{i=1}^{d} (x_i - \mu)^2}$ are mean and standard deviation, and γ and β are affine transform parameters. The calculation of μ only requires additions and multiplications by a constant. Thus, the main challenge lies in the required reciprocal square root operation. Next, we discuss the methods to cope with LayerNorm in different studies and report the performance in Tab. 10.

Some studies [7, 18, 64] have developed methods to mitigate the computation overhead of parameters μ and β for efficiency. For instance, THE-X [7] employs two sets of learnable parameters $\widetilde{\boldsymbol{\gamma}} := [\widetilde{\gamma}_1, \dots, \widetilde{\gamma}_d]$ and $\widetilde{\boldsymbol{\beta}} := [\widetilde{\beta}_1, \dots, \widetilde{\beta}_d]$ to avoid the calculations of original μ and β. Specifically, it simplifies LayerNorm as follows:

$$\text{LayerNorm}(\boldsymbol{x}) \approx \boldsymbol{x} \odot \boldsymbol{\gamma} + \boldsymbol{\beta}, \tag{24}$$

where $\widetilde{\boldsymbol{\gamma}}$ handles both normalization and linear transformation, and $\odot$ denotes for the Hadamard product. A distillation will be applied in (24) to learn from the original LN layers and bridge the performance gap. Iron [18] combines the weights of LayerNorm ($\frac{\gamma}{\sigma}$ and β) with the next linear layer's weights to save one matrix multiplication. This results in a simplified output from LayerNorm as follows:

$$\text{LayerNorm}(\boldsymbol{x}) \approx \left[x_i - \mu \mid_{i=1,\dots,d} \right]. \tag{25}$$

Table 10. A Comparison of LayerNorm Approximations.

Study	Input	Comm. (MB)	Comm. set.	Runtime (s)	Error	Comments
THE-X [7]	-	-	-	-	-	Require training
CipherGPT [20]	$\mathbb{R}^{256 \times 768}$	65	LAN[1]	4.756	-	-
SIGMA [17]	$\mathbb{R}^{128 \times 3072}$	112.64	LAN[2]	0.25	-	
PrivFormer [1]	-	1.64	LAN[3]/WAN[4]	0.077/10.753	-	
Iron [18]	$\mathbb{R}^{128 \times 768}$	871.46	LAN[1]/WAN[5]	16/1158	1.7×10^{-4}	
BOLT [43]	$\mathbb{R}^{128 \times 768}$	599.40	LAN[1]/WAN[5]	14/914	-	
NEXUS [64]	$\mathbb{R}^{128 \times 768}$	0	LAN[1]/WAN[5]	81/81	4.5×10^{-4}	Nearly comm. free

[1]3Gbps, 0.8ms   [2]9.4Gbps, 0.05ms   [3]4.93Gbps, 1.17ms   [4]97.4Mbps, 141.67ms   [5]100Mbps, 80ms

However, BOLT [43] points out that Iron ignores the residual connection in transformers and using Equa. (25) for LayerNorm would make the model's performance close to random guessing. Besides, NEXUS [64] performs the following transformation to LayerNorm :

$$\text{LayerNorm}(\boldsymbol{x}) = \left[\sqrt{d}\gamma \cdot \frac{z_i}{\sqrt{\sum_{i=1}^{d} z_i^2}} + \beta \,\bigg|_{i=1,\dots,d} \right], \tag{26}$$

where $z_i = dx_i - \sum_{i=1}^{d} x_i$. Then, NEXUS uses its proposed QuickSum function and the Newton iteration method in Qu et al. [45] to compute the summations and the inverse square root, respectively. Similar to NEXUS, PowerFormer [44] approximates $\frac{1}{\sqrt{x}}$ using an iterative algorithm based on Newton's method.

Some studies [1, 17, 20] employ customized protocols for the reciprocal square root operation in LayerNorm layers. CipherGPT [20] directly introduces an LUT to evaluate the reciprocal square root operation. Similarly, SIGMA [17] designs a custom 13-bit floating-point representation for efficiency to encode the input of LayerNorm and use it to index a LUT for the reciprocal square root. PrivFormer [1] utilizes the square-root inverse protocol in FALCON [29] for LayerNorm and further improves its security under MPC settings.

7 REPORTED EVALUATION

This section compares the experimental results presented in the selected studies.

7.1 Models and Datasets

The transformer models and datasets used for evaluating PTI performance in different studies are significant for comparison. For instance, transformers with more complex structures and a larger number of parameters tend to take longer to do inference, but they often perform better [23]. Below we will discuss popular models and datasets in PTI studies in detail.

Models. Popular transformer models for PTI studies [7, 18, 28, 34, 43, 64] mainly include the BERT family (e.g., Bert-Tiny [54], Bert-Base [10], Roberta-Base [31]) and the GPT family (e.g., GPT2-Base [46]). Some studies [12, 34] have also attempted to implement private inference on other transformer models, such as LLaMA-7B [53] and ViT-Base [13]. We present a detailed description of various transformer models in Table 11.

BERT's architecture is based on a multi-layer bidirectional Transformer encoder. It uses masked language modeling (MLM) and next sentence prediction (NSP) as pre-training objectives, which allow it to understand the context from

Table 11. Hyperparameters and Model Sizes of Various Transformer Models

Model	Layers (L)	Hidden size (d)	Heads (H)	MLP size	Params
BERT-Tiny [54]	2	128	2	512	4.4M
BERT-Base [10]	12	768	12	3072	110M
Roberta-Base [31]	12	768	12	3072	125M
BERT-Large [10]	24	1024	16	4096	340M
GPT2-Base [46]	12	768	12	3072	117M
GPT2-Medium [46]	24	1024	16	4096	345M
GPT2-Large [46]	36	1280	16	5120	762M
ViT-Base [13]	12	768	12	3072	86M
LLaMA-7B [53]	32	4096	32	11008	7B

Note: MLP means multi-layer perception, i.e., the feed-forward layers.

both directions (left and right of a token). This bidirectional understanding makes BERT especially effective for tasks requiring contextual inference, such as sentiment analysis, question answering, and language inference.

GPT models are based on a transformer decoder, a left-to-right architecture. They only predict future words based on past and current words, aligning them more closely with traditional language models. The primary training method is causal language modeling (CLM), where the model predicts the next word in a sentence, making GPT highly effective for tasks that involve generating coherent and contextually relevant text.

Evaluation Datasets. To assess the natural language understanding (NLU) abilities of different transformers, we need e general NLU benchmark for evaluation. Most PTI studies evaluate their performances on the GLUE benchmark [57], a widely adopted evaluation measure for BERT and GPT-based transformers. GLUE does not pose any constraints on model architecture beyond the ability to process single-sentence and sentence-pair inputs and make predictions, which makes it suitable for evaluating different transformers. Specifically, the GLUE benchmark contains three different kinds of NLU tasks and nine corresponding corpora: single-sentence tasks (CoLA, SST−2), similarity and paraphrase tasks (MRPC, STS−B, QQP), and inference tasks (MNLI, QNLI, RTE, WNLI).

Except for GLUE, some PTI studies [12, 34, 64] also use other benchmarks for evaluations. For instance, PUMA [12] and NEXUS [64] evaluate GPT2 on Wikitext-103 V1 [39] and CBT-CN, respectively. ViT-Base is particularly designed for image processing, and thus BumbleBee [34] uses the ImageNet dataset for its evaluation.

8 CONCLUSION

REFERENCES

[1] Yoshimasa Akimoto, Kazuto Fukuchi, Youhei Akimoto, and Jun Sakuma. 2023. Privformer: Privacy-preserving transformer with mpc. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 392−410.

[2] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. 2016. 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*. 805−817.

[3] Donald Beaver. 1992. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology—CRYPTO'91: Proceedings 11*. Springer, 420−432.

[4] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. 2018. Compressing vector OLE. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 896−912.

[5] Nishanth Chandran, Divya Gupta, Aseem Rastogi, Rahul Sharma, and Shardul Tripathi. 2019. EzPC: Programmable and efficient secure two-party computation for machine learning. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 496−511.

[6] Dake Chen, Yuke Zhang, Souvik Kundu, Chenghao Li, and Peter A Beerel. 2023. RNA-ViT: Reduced-Dimension Approximate Normalized Attention Vision Transformers for Latency Efficient Private Inference. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE,

Table 12. Resource requirements, Tasks performed, Dataset Performance, and Runtime.

Study	Model	Dataset	I./O. size	Comm.	Comm. Set.	Performance			Runtime
						Plain	Enc.	Loss ↓	
MPCFormer [28]	BERT-Base	QNLI MRPC RTE STS-B	(128,1)	12.089 GB	(5 Gbps, 1 ms)	91.7 % 90.3 % 69.7 % 89.1 %	90.6 % 88.7 % 64.9 % 80.3 %	1.1 % 1.6 % 4.8 % 8.8 %	55.320 s
THE-X [7]	BERT-Tiny	SST-2 MRPC RTE STS-B	-	-	-	82.45 % 81.57 % 58.56 % 72.83 %	82.11 % 79.94 % 58.12 % 68.39 %	0.34 % 1.63 % 0.44 % 4.44 %	-
PUMA [12]	BERT-Base	CoLA RTE QNLI	(128,1)	10.773 GB	(5 Gbps, 1 ms)	61.6 % 70.0 % 91.6 %	61.3 % 70.0 % 91.6 %	0.3 % 0.0 % 0.0 %	33.913 s
	Roberta-Base	-		11.463 GB		-	-	-	41.641 s
	Bert-Large	-		27.246 GB		-	-	-	73.720 s
	GPT2-Base	Wiki.-103	(32,1)	3.774 GB		16.284	16.284	0.000	15.506 s
	GPT2-Medium			7.059 GB		12.536	12.540	-0.004	30.272 s
	GPT2-Large			11.952 GB		10.142	10.161	-0.019	54.154 s
	LLaMA-7B	-	(8,1)	1.794 GB		-	-	-	200.473 s
Iron [18]	BERT-Base	SST-2 MRPC RTE STS-B	(128,1)	280.99 GB	(3 Gbps, 0.8 ms)	92.36 % 90.00 % 69.70 % 89.62 %	92.77 % 89.87 % 70.76 % 89.41 %	-0.41 % 0.13 % -1.06 % 0.21 %	475 s
BumbleBee [34]	BERT-Base	QNLI RTE CoLA	(128,1)	-	(1 Gbps, 0.5 ms)	90.30 % 70.04 % 61.57 %	90.20 % 70.04 % 60.82 %	0.10 % 0.00 % 0.75 %	-
	BERT-Large	-		20.85 GB		-	-	-	404.4 s
	LLaMA-7B	-		6.82 GB		-	-	-	832.2 s
	GPT2-Base	-	(32,1)	1.94 GB		-	-	-	55.2 s
	ViT-Base	ImageNet	([224,224,3],1)	14.44 GB		89.44 %	89.13 %	0.31 %	234 s
BOLT [43]	BERT-Base	SST-2 MRPC RTE STS-B	-	25.74 GB	(3 Gbps, 0.8 ms)	92.36 % 90.00 % 69.70 % 89.62 %	92.78 % 89.95 % 69.31 % 88.44 %	-0.42 % 0.05 % 0.39 % 1.18 %	185 s
NEXUS [64]	BERT-Base	RTE SST-2 QNLI	(128,1)	0.16 GB	(100 Mbps, 80 ms)	70.04 % 92.36 % 90.30 %	69.88 % 92.11 % 89.92 %	0.16 % 0.25 % 0.38 %	1125 s
	GPT2-Base	CBT-CN	-	-	-	85.70 %	85.31 %	0.39 %	-

1–9.

[7] Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. The-x: Privacy-preserving transformer inference with homomorphic encryption. *arXiv preprint arXiv:2206.00216* (2022).

[8] Yuntian Chen, Xianjia Meng, Zhiying Shi, Zhiyuan Ning, and Jingzhi Lin. 2024. SecureTLM: Private inference for transformer-based large model with MPC. *Information Sciences* 667 (2024), 120429.

[9] Edward Chou, Josh Beal, Daniel Levy, Serena Yeung, Albert Haque, and Li Fei-Fei. 2018. Faster cryptonets: Leveraging sparsity for real-world encrypted inference. *arXiv preprint arXiv:1811.09953* (2018).

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] Yuanchao Ding, Hua Guo, Yewei Guan, Weixin Liu, Jiarong Huo, Zhenyu Guan, and Xiyong Zhang. 2023. East: Efficient and accurate secure transformer framework for inference. *arXiv preprint arXiv:2308.09923* (2023).

[12] Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, and Wenguang Cheng. 2023. Puma: Secure inference of llama-7b in five minutes. *arXiv preprint arXiv:2307.12533* (2023).

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[14] David Evans, Vladimir Kolesnikov, Mike Rosulek, et al. 2018. A pragmatic introduction to secure multi-party computation. *Foundations and Trends®*
     in Privacy and Security 2, 2-3 (2018), 70–246.

[15] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of*
     computing. 169–178.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[17] Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. 2023. SIGMA: secure GPT
     inference with function secret sharing. *Cryptology ePrint Archive* (2023).

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

[19] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18, 7 (2006),
     1527–1554.

[20] Xiaoyang Hou, Jian Liu, Jingyu Li, Yuhan Li, Wen-jie Lu, Cheng Hong, and Kui Ren. 2023. Ciphergpt: Secure two-party gpt inference. *Cryptology*
     ePrint Archive (2023).

[21] Hai Huang and Yongjian Wang. 2024. SecBERT: Privacy-preserving pre-training based neural network inference system. *Neural Networks* 172
     (2024), 106135.

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

[23] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language
     understanding. *arXiv preprint arXiv:1909.10351* (2019).

[24] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. {GAZELLE}: A low latency framework for secure neural network
     inference. In *27th USENIX security symposium (USENIX security 18)*. 1651–1669.

[25] Marcel Keller. 2020. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC conference on computer*
     and communications security. 1575–1590.

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

[27] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2020. Cryptflow: Secure tensorflow inference.
     In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 336–353.

[28] Dacheng Li, Hongyi Wang, Rulin Shao, Han Guo, Eric Xing, and Hao Zhang. 2022. MPCFORMER: FAST, PERFORMANT AND PRIVATE TRANS-
     FORMER INFERENCE WITH MPC. In *The Eleventh International Conference on Learning Representations*.

[29] Shaohua Li, Kaiping Xue, Bin Zhu, Chenkai Ding, Xindi Gao, David Wei, and Tao Wan. 2020. Falcon: A fourier transform based approach for fast and
     secure convolutional neural network predictions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8705–8714.

[30] Xuanqi Liu and Zhuotao Liu. 2023. Llms can understand encrypted prompt: Towards privacy-computing friendly transformers. *arXiv preprint*
     arXiv:2305.18396 (2023).

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

[32] Yanxin Liu and Qianqian Su. 2024. PPTIF: Privacy-Preserving Transformer Inference Framework for Language Translation. *IEEE Access* (2024).

[33] Natasha Lomas. 2023. Italy orders ChatGPT blocked citing data protection concerns. *TechCrunch, March* 31 (2023).

[34] Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Kui Ren, Cheng Hong, Tao Wei, and WenGuang Chen. 2023. Bumblebee: Secure two-party
     inference framework for large transformers. *Cryptology ePrint Archive* (2023).

[35] Brady D Lund and Ting Wang. 2023. Chatting about ChatGPT: how may AI and GPT impact academia and libraries? *Library hi tech news* 40, 3
     (2023), 26–29.

[36] Jinglong Luo, Yehong Zhang, Zhuo Zhang, Jiaqi Zhang, Xin Mu, Hui Wang, Yue Yu, and Zenglin Xu. 2024. SecFormer: Fast and Accurate
     Privacy-Preserving Inference for Transformer Models via SMPC. In *Findings of the Association for Computational Linguistics ACL 2024*. 13333–13348.

[37] Junming Ma, Yancheng Zheng, Jun Feng, Derun Zhao, Haoqi Wu, Wenjing Fang, Jin Tan, Chaofan Yu, Benyu Zhang, and Lei Wang. 2023. {SecretFlow-
     SPU}: A Performant and {User-Friendly} Framework for {Privacy-Preserving} Machine Learning. In *2023 USENIX Annual Technical Conference*
     (USENIX ATC 23). 17–33.

[38] Cecily Mauran. 2023. Whoops, Samsung workers accidentally leaked trade secrets via ChatGPT. *Mashable [online]. Dostupné z: https://mashable.*
     com/article/samsungchatgpt-leak-details (2023).

[39] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* (2016).

[40] Microsoft and OpenAI. 2023. Bing Chat. (2023). https://www.bing.com/search

[41] Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. 2024. THOR: Secure Transformer Inference with Homomorphic Encryption. *Cryptology*
     ePrint Archive (2024).

[42] OpenAI. 2022. ChatGPT. (2022). https://openai.com/blog/chatgpt

[43] Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. 2024. BOLT: Privacy-Preserving, Accurate and Efficient Inference
     for Transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 130–130.

[44] Dongjin Park, Eunsang Lee, and Joon-Woo Lee. 2024. Powerformer: Efficient privacy-preserving transformer with batch rectifier-power max function and optimized homomorphic attention. *Cryptology ePrint Archive* (2024).

[45] Hongyuan Qu and Guangwu Xu. 2023. Improvements of Homomorphic Secure Evaluation of Inverse Square Root. In *International Conference on Information and Communications Security*. Springer, 110–127.

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

[47] Deevashwer Rathee, Dacheng Li, Ion Stoica, Hao Zhang, and Raluca Popa. 2024. MPC-Minimized Secure LLM Inference. *arXiv preprint arXiv:2408.03561* (2024).

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

[49] Lorenzo Rovida and Alberto Leporati. 2024. Transformer-based language models and homomorphic encryption: An intersection with bert-tiny. In *Proceedings of the 10th ACM International Workshop on Security and Privacy Analytics*. 3–13.

[50] Manuel B Santos, Dimitris Mouris, Mehmet Ugurbil, Stanislaw Jarecki, José Reis, Shubho Sengupta, and Miguel de Vega. 2024. Curl: Private LLMs through Wavelet-Encoded Look-Up Tables. *Cryptology ePrint Archive* (2024).

[51] Sijun Tan, Brian Knott, Yuan Tian, and David J Wu. 2021. CryptGPU: Fast privacy-preserving machine learning on the GPU. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1021–1038.

[52] Ahmed Tlili, Boulus Shehata, Michael Agyemang Adarkwah, Aras Bozkurt, Daniel T Hickey, Ronghuai Huang, and Brighter Agyemang. 2023. What if the devil is my guardian angel: ChatGPT as a case study of using chatbots in education. *Smart learning environments* 10, 1 (2023), 15.

[53] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[54] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962* (2019).

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[56] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. 2021. FALCON: Honest-Majority Maliciously Secure Framework for Private Deep Learning. *Proceedings on Privacy Enhancing Technologies*.

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

[58] Weize Wang and Yi Kuang. 2024. CipherFormer: Efficient Transformer Private Inference with Low Round Complexity. *arXiv preprint arXiv:2403.16860* (2024).

[59] Yongqin Wang, G Edward Suh, Wenjie Xiong, Benjamin Lefaudeux, Brian Knott, Murali Annavaram, and Hsien-Hsin S Lee. 2022. Characterization of mpc-based private inference for transformer-based models. In *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 187–197.

[60] Tianshi Xu, Lemeng Wu, Runsheng Wang, and Meng Li. 2024. PrivCirNet: Efficient Private Inference via Block Circulant Transformation. *arXiv preprint arXiv:2405.14569* (2024).

[61] Andrew C Yao. 1982. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 160–164.

[62] Chenkai Zeng, Debiao He, Qi Feng, Xiaolin Yang, and Qingcai Luo. 2024. SecureGPT: A Framework for Multi-Party Privacy-Preserving Transformer Inference in GPT. *IEEE Transactions on Information Forensics and Security* (2024).

[63] Wenxuan Zeng, Meng Li, Wenjie Xiong, Tong Tong, Wen-jie Lu, Jin Tan, Runsheng Wang, and Ru Huang. 2023. Mpcvit: Searching for accurate and efficient mpc-friendly vision transformer with heterogeneous attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5052–5063.

[64] Jiawen Zhang, Jian Liu, Xinpeng Yang, Yinghao Wang, Kejia Chen, Xiaoyang Hou, Kui Ren, and Xiaohu Yang. 2024. Secure Transformer Inference Made Non-interactive. *Cryptology ePrint Archive* (2024).

[65] Yuke Zhang, Dake Chen, Souvik Kundu, Chenghao Li, and Peter A Beerel. 2023. Sal-vit: Towards latency efficiefnt private inference on vit using selective attention search with a learnable softmax approximation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5116–5125.

[66] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yu-an Tan. 2019. Secure multi-party computation: theory, practice and applications. *Information Sciences* 476 (2019), 357–372.

[67] Mengxin Zheng, Qian Lou, and Lei Jiang. 2023. Primer: Fast private transformer inference on encrypted data. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[68] Itamar Zimerman, Allon Adir, Ehud Aharoni, Matan Avitan, Moran Baruch, Nir Drucker, Jenny Lerner, Ramy Masalha, Reut Meiri, and Omri Soceanu. 2024. Power-Softmax: Towards Secure LLM Inference over Encrypted Data. *arXiv preprint arXiv:2410.09457* (2024).

[69] Itamar Zimerman, Moran Baruch, Nir Drucker, Gilad Ezov, Omri Soceanu, and Lior Wolf. 2023. Converting transformers to polynomial form for secure inference over homomorphic encryption. *arXiv preprint arXiv:2311.08610* (2023).

A MPC SETTINGS

We first present the popular MPC settings employed in current secure inference studies. Each of them has different requirements and assumptions on the participating parties. Generally speaking, the 2PC setup is the most natural for secure inference, while 3PC and 2PC-Dealer setups usually support a wider variety of operations and improve the MPC performance. We compare their strengths and weaknesses in Table. 13.

- **2-party computation (2PC):** This is the basic MPC setting where two participants engage in computation without mutual trust. Each party contributes their input to the computation, but neither can see the other's data. This setting is the most intuitive and commonly used for secure inference, as it only involves two parties and minimizes the complexity of trust and coordination.
- **Honest-majority 3-party computation (3PC):** This configuration introduces an additional "helper" party, increasing the capabilities of the computation. In this model, the adversary may corrupt any single party without compromising the integrity of the computation. The presence of the third party allows for a broader range of operations and enhances the overall performance of MPC.
- **2PC with trusted dealer (2PC-Dealer):** In this scenario, a trusted dealer plays a crucial role during a pre-processing phase by distributing *input-independent* correlated randomness to the two computation parties. This randomness is used to facilitate and speed up computations. The trusted dealer is not involved in the computation itself but enables more efficient processing.

Table 13. A Comparison of secure MPC schemes with respect to different key characteristics.

Scheme	#Parties	Security Model	Overhead		Resilience	Practicality	Scalability
			Comm.[1]	Comp.[2]			
2PC	2	-	+			+	+
3PC	3	Honest-majority	-	+		-	++
2PC-Dealer	3	Trusted Dealer		+			++

[1] Communication.   [2] Computation.
Note that the number of '+' indicates the degree of **benefit** with respect to a feature, with '+++' denoting the strongest and '+' representing the weakest.