Sok: Private Transformer-Based Model Inference

Yuntian Chen[1,2,6], Tianpei Lu[3,4], Zhanyong Tang[1,2,5,✉], Bingsheng Zhang[3,4], Zhiying Shi[1,2,5], Yuxiang Luan[1,2] *and* Zhuzhu Wang[1,2,6]

[1]*Northwest University, {chenyt, shizhiying,luanxx}@stumail.nwu.edu.cn, {zytang, zzwang}@nwu.edu.cn,*

[2]*Xi'an Key Laboratory of Advanced Computing and Software Security*

[3]*The State Key Laboratory of Blockchain and Data Security, Zhejiang University, {lutianpei, bingsheng}@zju.edu.cn*

[4]*Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security*

[5]*Shaanxi Key Laboratory of Passive Internet of Things and Neural Computing*

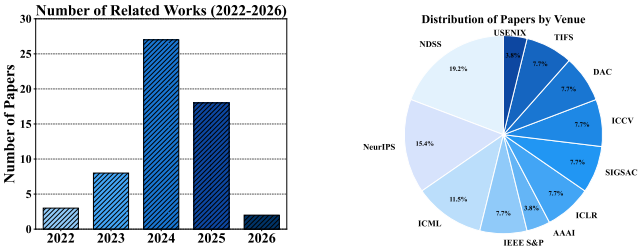
[6]*Shaanxi International Joint Research Centre for the Battery-Free Internet of Things*

Abstract

The growing demand for privacy-preserving Transformer inference has led to the emergence of numerous protocols designed to protect sensitive data and model parameters. These protocols utilize diverse cryptographic tools under varying assumptions, each presenting unique characteristics and trade-offs between computation, communication, and accuracy. In this paper, we conduct a systematic and in-depth analysis of existing approaches from diverse performance perspectives, identifying their limitations and research gaps. We further evaluate the reproducibility of prior systems and re-benchmark representative solutions under standardized configurations. Our results yield a principled guideline for balancing protocol trade-offs under different deployment settings.

1 Introduction

Transformer-based models [1], including large language models [2], vision language models [3], and other multimodal foundation models [4], are being widely deployed in privacy-sensitive scenarios [5]. In particular, these frameworks have found critical domain-specific applications in areas such as private text generation [6] and secure biomedical image analysis with transformers [7], where protecting proprietary inputs and patient confidentiality is paramount. Consequently, there is an urgent demand for privacy-preserving computation. However, model providers and cloud platforms are not always fully trustworthy, and users' inference queries and associated data are therefore exposed to potential privacy leakage risks [8]. Although cryptographic techniques can, in principle, support a variety of privacy services, private transformer-based model inference remains highly challenging due to the model scale, computational complexity, and communication overhead. While existing solutions have demonstrated the feasibility of supporting non-real-time tasks (e.g., private long-form text generation), they still fall short in terms of efficiency and practicality for latency-sensitive and security-critical real-time interactive scenarios [9].



(a) Annual Publication Trend (b) Venue Distribution

Figure 1: Statistical analysis of private transformer-based model inference research in recent years (2022–2025).

In practice, transformer-based models are commonly deployed in outsourced settings, which makes privacy-preserving inference for Transformer models a highly immediate and practically relevant problem. We begin by examining research trends in this area and then follow a top-down analysis, moving from outsourced deployment models that affect privacy, to the cryptographic techniques used in prior work, and finally to concrete performance comparisons.

Dissecting the Development. Research on privacy-preserving inference for transformer models has developed at the intersection of deep learning, cryptography, and systems, posing a range of challenging problems. We systematically reviewed over 61 frameworks from the past years, analyzed their technical contributions in depth, and categorized them based on problem settings and primary methodologies (see Table 2). Figure 1 illustrates the research lineage, helping researchers identify gaps and potential opportunities.

C/S Setting. This refers to a two-party computation setting in which the model owner (server) and the querier (client) jointly execute the inference task. The security objectives are twofold: to protect the user's query data privacy and to preserve the model's intellectual property. In this setting, both parties are active participants in the computation, without

✉ Corresponding author: zytang@nwu.edu.cn

Table 1: Enhanced Capability Comparison of Our SoK and Prior Works on Private Inference

Work	Transformer	HE	MPC	FSS	Hybrid	Protocol Conv.	Code Survey	Systematic Repro. & Eval.	Extensibility Survey
Wang et al. (ISPASS 2022) [18]	✓	✗	✓	✗	✗	✗	✗	✓	✗
Lucien et al. (S&P 2023) [10]	✗	✓	✓	✓	✓	✗	✗	✓	✗
Martin (ePrint 2025) [19]	✓	✓	✓	✓	✓	✓	✗	✗	✗
Zeng et al. (arXiv 2025) [9]	✓	✓	✓	✓	✓	✓	✓	✗	✗
Luo et al. (Artif Intell Rev 2025) [20]	✓	✓	✓	✓	✓	✗	✓	✗	✗
Davide et al. (ePrint 2026) [21]	✓	✓	✓	✓	✓	✓	✗	✗	✓
This Work	✓	✓	✓	✓	✓	✓	✓	✓	✓

Notation. ✓ indicates that the corresponding topic is systematically analyzed or serves as a core component of the work. ✓indicates that the topic is mentioned but not analyzed in depth. ✗ indicates that the topic is not covered.

relying on any additional third parties [10].

Outsourcing Setting. Given the high computational cost of large models, model providers often cannot perform complete inference locally. Through secure outsourced inference, users can delegate inference requests to untrusted third-party servers while ensuring that the model parameters remain confidential. This approach is the primary mode of privacy-preserving inference for large models in practice and also allows for feasible user-side involvement in the computation [11].

Early Works. Early privacy-preserving inference studies on Transformer models (e.g., BERT-Base) primarily focused on protocol-level designs based on a single cryptographic primitive. In 2022, THE-X [12] pioneered Homomorphic encryption (HE)-based Transformer inference by computing linear layers homomorphically on the server while delegating nonlinear layers to the client for local plaintext evaluation and re-encryption. In contrast, Secure Multi-party Computation (MPC)-based systems such as Privformer [13] and PUMA [14] realize end-to-end secure inference using three-party computation (3-PC), where nonlinear layers are handled via function substitution or polynomial approximation. Subsequent efforts explored model-level optimizations to reduce secure computation cost, including quantization-aware distillation (e.g., MPCFormer [15]) and MPC-aware quantization schemes (e.g., Ditto [16]). At the system level, recent works further improve efficiency by optimizing the execution stack, most notably through GPU acceleration [17].

Performance Comparisons. This SoK explains the breakthroughs of existing transformer-based privacy-preserving inference schemes and common benchmarking methods. We also clarify how to evaluate frameworks, select appropriate approaches for different scenarios, and conduct fair comparisons. For instance, frameworks that rely on model quantization should not be directly compared with those that do not. Moreover, we re-evaluate state-of-the-art frameworks that were originally designed for wide-area network (WAN) settings, and observe that under certain network configurations, some schemes fail to reach the performance suggested by their protocol designs. While such schemes may be unsuitable for latency-sensitive real-time interactions, they remain practical for non-real-time application scenarios.

Fostering Interdisciplinary Research. Many issues in private transformer-based model inference may appear complex to external researchers. We explain these challenges and mitigation techniques, and classify the technical contributions based on Table 8 in the Appendix B, allowing experts from different domains to locate relevant studies quickly, focus on the core issues, and avoid redundant efforts. At the end of each subsection, we also highlight key challenges and under-explored open problems.

Scope. This survey focuses on cryptography-based methods, analyzing transformer-based privacy-preserving inference from model, protocol, and system perspectives. It does not cover trusted execution environment (TEE) solutions [22], membership inference attacks [23], model inversion attacks [24], or federated learning [25]. Following this scope, we systematically selected the 61 cryptography-based works surveyed in this paper, all of which propose concrete private inference protocols or systems for Transformer models.

Literature collation. To ensure a systematic overview, we selected the 61 cryptography-based works through a multi-stage literature search and filtering process. First, we leveraged Google Scholar, DBLP, and IEEE Xplore to source literature published up to 2026. Our search targeted top-tier venues in security, systems, and machine learning (as summarized in Figure **??**). Then, we executed queries combining cryptographic primitives and model architectures, including: *"Private Inference"*, *"Secure Machine Learning Inference"*, *"Fully Homomorphic Encryption"*, *"Multi-party Computation"*, *"Transformer"*, and *"Privacy Preserving"*. Finally, we filtered the results to include only cryptographic protocols or frameworks optimized for transformer-based private inference, resulting in the 61 cornerstone works analyzed in **SoK**.

Motivation. This work goes beyond a simple summary of existing studies. It provides a systematic and in-depth analysis of prior approaches, identifies their limitations and gaps, and offers new insights (see Table 1). As summarized in Table 4, we examine existing surveys on privacy-preserving inference. However, rather than merely cataloging prior work, we emphasize systematic comparison and analysis to uncover the intrinsic differences and limitations. Notably, although some prior studies focus on general privacy-preserving machine learning

(PPML) tasks and include transformer-based models as part of their scope, our investigation reveals that among 54 PPML works, only six (≈11%) can be directly extended to private transformer-based model inference (see Section 8.3). This observation highlights the necessity of a dedicated analysis of private transformer-based model inference. As the field is rapidly evolving, we also maintain a public GitHub repository to continuously track new developments, available at https://anonymous.4open.science/r/SoK4PPLLM (Or Zenodo repository at https://zenodo.org/records/20319561). We believe this dynamic component further enhances the utility and longevity of our contribution.

Roadmap: Section 2 is the preliminary. Section 3 is the overview. Section 4, 5, 6 present frameworks based on HE, MPC (ASS, FSS, RSS), and hybrid HE–MPC, respectively. Section 7 presents model optimization techniques. Section 8 examines practical deployment considerations, including hardware acceleration, system- and compiler-level optimizations. Section 9 synthesizes the performance of existing frameworks with respect to accuracy, latency, and system adaptability.

2 Preliminary

2.1 Transformer-Based Models

Transformer models learn sequence representations via self-attention and can be abstracted as a stack of attention-based and feed-forward layers. This abstraction covers common Transformer variants such as BERT [26], GPT [27], and LLaMA [28]. Appendix A details the computation of the model using BERT [26] as an example, with the architecture shown in Figure 6. Modern machine-learning frameworks such as Caffe [29], Keras, PyTorch, TensorFlow [30], and Theano make it straightforward to implement and deploy large models based on the Transformer architecture, including BERT [26] and its variants. Built on top of these frameworks, specialized libraries such as the Hugging Face ecosystem [31] further streamline the loading, composition, and invocation of transformer pre-trained models through unified model definitions and checkpoint management. To provide a comprehensive overview, Table 13 in Appendix B summarizes all transformer-based models covered in the SoK.

Linear Layers. Typical linear layers include fully connected linear mappings as well as projection operations implemented via matrix multiplication, such as the Query, Key, and Value projections in the attention mechanism and the linear transformations in the feed-forward network (FFN) [1]. Formally, the output of a linear layer can be expressed as $y_i = \sum_j w_{ij} \cdot x_j$, where w_{ij} denotes the weight parameter and x_j denotes the input. From this perspective, the evaluation of a linear layer can be reduced to a sequence of dot-product computations.

Non-Linear Layers. In transformer-based models (e.g., BERT [26] and its variants), non-linear computations are essential for representation learning and often dominate the cost of privacy-preserving inference. i) Activation functions apply element-wise non-linearities in feed-forward networks (FFNs), such as ReLU or GeLU, while SoftMax is widely used in self-attention and output layers. ii) Pooling and aggregation operations produce global representations from token-level outputs, typically via sequence-level reductions such as mean pooling or special tokens (e.g., `[CLS]` [32]), rather than explicit spatial pooling. iii) Normalization layers stabilize intermediate activations, with Transformer models predominantly adopting layer normalization (LayerNorm) along the feature dimension. iv) Output layers generate final predictions from the last hidden representations, commonly involving SoftMax or argmax.

Common Benchmarks. Private inference for transformer-based models aims to reduce latency, i.e., the time between issuing a query and receiving the result. Another key metric is communication overhead, which measures the amount of data that must be transmitted by the protocol within a given time. In addition, inference accuracy is a crucial consideration, as the protocols often introduce approximations (e.g., fixed-point representations or polynomial approximations of nonlinear functions [33]). All three-latency, communication overhead, and inference accuracy-are jointly affected by the choice of secure protocols, large model architectures, network conditions, and hardware components.

GPUs are highly optimized for floating-point arithmetic and thus constitute the dominant hardware platform for neural network computation. In contrast, cryptographic primitives mainly rely on fixed-point or large-integer arithmetic, which does not naturally align with the GPU computation model, making efficient GPU utilization for privacy-preserving neural network inference challenging [34]. Nevertheless, recent work has started to explore hardware acceleration for cryptographic workloads, especially homomorphic encryption, via GPU-based techniques or specialized accelerators to reduce the performance gap between private and plaintext computation [35].

2.2 Problem Formulation and Threat Models

The C/S setting can be formalized as a secure two-party computation: $f(\text{model}, \text{query}) \rightarrow (\text{result})$, where the model owner and the querier are referred to as the *client* and *server*, respectively. This paper adopts the client/server (C/S) notation to avoid confusion with generic two-party computation (2-PC) frameworks, while abstracting away the details of auxiliary service providers. In certain C/S scenarios, a dealer may be introduced as an auxiliary node to generate or distribute secret shares or pre-computed materials [36]. Outsourcing Setting can be viewed as a specific instance of this 2-PC (or n-party, n-PC) setting, in which the computation is "outsourced" to the server or auxiliary nodes without revealing the client's query or the model's parameters. Such an outsourcing setting may also involve three-party or multi-party setups, under the

assumption that the honest parties' hardware and software are not compromised, that is, the parties do not collude. The final result is safely delivered solely to the querier and kept strictly private from the server or outsourcing providers.

Frameworks typically consider two types of adversaries: *Semi-honest*, who follow the protocol but attempt to learn additional information, and *malicious*, who may arbitrarily deviate from the protocol. Among the 61 works we surveyed, the vast majority assume a Semi-honest model, with only two studies targeting malicious adversaries. This persistent gap stems from the substantial latency overhead introduced by cryptographic verification mechanisms like ZKPs or MACs, which challenges the deployment of real-time inference. Furthermore, these malicious-secure primitives require complex algebraic operations over large fields, which are difficult to pipeline efficiently on standard ML hardware like GPUs. Consequently, until hardware acceleration or co-design frameworks mitigate these overheads, scaling malicious security to practical Transformer deployment will remain a major bottleneck.

2.3 Cryptographic Primitives

This section covers key cryptographic primitives for private Transformer inference.

MPC [37] allows mutually distrustful parties to jointly compute a known function over their respective secret inputs via interactive protocols. The computation results are often secret-shared among the participants.

i). Additive Secret Sharing (ASS) [38] is a fundamental building block of MPC. A secret-shared value is commonly denoted as $\langle x \rangle$, depending on the underlying sharing scheme. In 2-out-of-2 arithmetic secret sharing, a value $x \in \mathbb{Z}_{2^\ell}$ is additively shared as $x = \langle x \rangle_0 + \langle x \rangle_1 \pmod{2^\ell}$, where party P_i holds share $\langle x \rangle_i$.

ii). Boolean Secret Sharing (BSS) [39] operates over the ring $\mathbb{Z}_2$. A bit-vector $x \in \{0,1\}^\ell$ is shared as $x = \langle x \rangle_0^{\mathsf{B}} \oplus \langle x \rangle_1^{\mathsf{B}}$, where $\oplus$ denotes bit-wise XOR and party P_i holds share $\langle x \rangle_i^{\mathsf{B}}$.

iii). Function Secret Sharing (FSS) [40] generalizes secret sharing from values to functions. An FSS scheme consists of two algorithms $(\mathsf{Gen}, \mathsf{Eval})$. Given a function description $\hat{g}$, the key generation algorithm produces function shares $(k_0, k_1) \leftarrow \mathsf{Gen}(1^\lambda, \hat{g})$, such that for any input x, $\mathsf{Eval}(0, k_0, x) + \mathsf{Eval}(1, k_1, x) = g(x)$. Each party learns only its function share, which reveals no information about the underlying function g.

iv). Replicate Secret Sharing (RSS) [41] is widely used in honest-majority MPC, especially for three-party computation. In 2-out-of-3 replicated secret sharing, a secret $x \in \mathbb{Z}_{2^\ell}$ is split into three random shares x_0, x_1, x_2 such that $x = x_0 + x_1 + x_2 \pmod{2^\ell}$, distributed to parties P_0, P_1, P_2 with P_i holding (x_i, x_{i+1}).

v). Yao's Garbled Circuit (GC) [42] enables two-party computation (2PC) over Boolean circuits. The Garbler sends garbled gates (tables) to the Evaluator, who obtains garbled inputs via Oblivious Transfer (OT). Gates are evaluated in topological order, producing secret-shared output.

vi). Oblivious Transfer (OT) [43], [44] is a fundamental primitive for secure selection of private data. In a (k, n)-OT, the sender holds n messages of k bits and the receiver inputs an index $i \in [n]$, obtaining the i-th message without learning others, while the sender learns nothing about i.

Homomorphic Encryption (HE) [45] enables computation directly on encrypted data. For a plaintext x, its ciphertext is denoted by $[\![x]\!] = \mathsf{Enc}_{pk}(x)$, which can be correctly decrypted using the matching secret key sk. Most practical systems considered in this SoK adopt *Fully Homomorphic Encryption (FHE)*, which supports a bounded number of homomorphic multiplications determined by the arithmetic circuit depth. This bound arises from noise accumulation during homomorphic operations, which may eventually cause decryption failure. To manage noise growth, bootstrapping can be employed. However, its high computational complexity remains a primary bottleneck.

Representative FHE schemes include BFV [46], BGV [47], and CKKS [48], which are also the most widely used FHE schemes in private transformer-based model inference. They support SIMD execution, packing multiple plaintext slots into a single ciphertext and applying operations slot-wise in parallel. For vectors $(x_i)_i$ and $(y_i)_i$, homomorphic operations satisfy $\mathsf{Dec}([\![(x_i)_i]\!] \star [\![(y_i)_i]\!]) = (x_i \star y_i)_i$, where $\star$ denotes addition or multiplication. Despite sharing the same SIMD computation model, these schemes differ in their supported numeric semantics. BFV and BGV support exact integer arithmetic, whereas CKKS enables approximate floating-point computation with rescaling for scale control, making CKKS the preferred choice in practice. FHE ciphertext-rotation cyclically shifts plaintext slots in an encrypted vector; it is costly in HE and a key bottleneck for matrix multiplication. In contrast, TFHE [49] is an LWE-based FHE scheme featuring fast bootstrapping and efficient Boolean gate evaluation, but its arithmetic operations are significantly more expensive, which limits its applicability to arithmetic-intensive neural network workloads. Without loss of generality, in the following, we use "HE" to refer collectively to FHE, and related methods, but not including TFHE.

Comparison of the Gadgets. ASS is highly efficient for addition and multiplication: additions are performed locally with only a few CPU instructions, while each multiplication requires one round of communication. As a result, ASS is well-suited for linear layers, but its performance may be constrained by communication overhead in high-latency networks. In contrast, HE eliminates interaction during computation, making it more favorable in high-latency or communication-free settings, which also motivates the design of pure HE frameworks (Sections 4). Boolean- or circuit-based primitives (e.g., TFHE, BSS, and Yao's GC) theoretically support the evaluation of arbitrary functions and are

particularly suitable for nonlinear operations such as comparisons and activation functions. FSS encodes functions as distributed shares to reduce online communication overhead, but it typically relies on an additional third party (e.g., a dealer), and only a limited number of works have explored its application to large-model inference [50]. RSS, under three-party or honest-majority settings, enables efficient linear computation through local additions and limited interaction, and generally incurs a lower communication cost than standard arithmetic secret sharing.

Overall, in transformer-based model inference, arithmetic secret sharing, RSS, and homomorphic encryption remain the primary tools for linear layers. Although BSS, Yao's GC, and TFHE support nonlinear computation, their high computational cost limits their direct use in practice, where approximations are often adopted instead. Recent work increasingly favors hybrid protocols that select appropriate primitives based on operation types and rely on a small number of protocol conversions to achieve efficient and flexible privacy-preserving inference.

Fixed-point Representation. [51] Most cryptographic primitives operate over finite fields or rings. For example, standard ASS uses $\mathbb{Z}_{2^\ell}$, while HE typically uses a large prime modulus $\mathbb{Z}_p$. Neural networks, however, use floating-point numbers. A float x_f can be quantized as $x_Q = x_f \cdot S_x$, where S_x is a scaling factor proportional to the protocol's accuracy. Computations like dot products are performed in the fixed-point domain to prevent overflow, and the result y_Q is dequantized via $y_f = y_Q / S_y$.

Bitwidth is defined as the number of bits allocated to represent operands, and it has a direct impact on computational efficiency. In the context of MPC, an increased bitwidth enlarges data-transfer volumes, raises the number of CPU/GPU instructions, and expands the size of the underlying circuits, thereby elevating both communication and computation overheads.

3 Overview

This SoK classifies private Transformer inference works into three categories according to their core cryptographic techniques: HE-based, MPC-based, and HE-MPC hybrid. Table 2 summarizes the deployment scenarios supported by each type. Building on this, this section first presents key performance milestones to demonstrate recent progress in the field.

Cryptographic operations typically incur high overhead and often suffer from reduced precision when operating over fixed-point numbers (e.g., see Section 7). Here, we highlight early challenges and some milestones. Performing a single private inference on 128 tokens in a BERT-based model using THE-X [12] takes roughly **78** minutes (from [52]), respectively. Two years later, BOLT [33] reduces this to **369.19** seconds (from [53]). To date, leveraging hybrid-framework approaches, SIGMA [50] achieves a runtime of just **13.99** seconds (measured in our own experiments on an A10 GPU).

Table 2: Privacy-Preserving Deployment Scenarios Supported by Different Framework Types.

Framework Type	C/S Setting	Outsourcing Setting	C/S Setting with Dealer
HE (§4)	●	◐	○
MPC (§5)	●	●	◐
Hybrid (§6)	●	◐	○

Notes: ● : ALL (Frameworks of the type support the service), Some: ◐ , None: ○

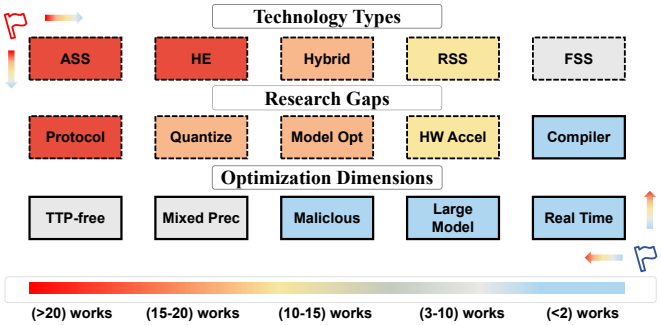


Figure 2: The ▭ indicates the dimensions addressed in prior studies, whereas the ▭ designates the dimensions emphasized in the present work. Similarly, the ⚑ highlights the principal focus of the current research landscape, while the ⚑ marks those directions that remain relatively under-explored and therefore merit further investigation in future studies.

Researchers in private transformer-based model inference analyze operations across multiple levels and tailor protocols under constraints such as CPU/GPU environments, accuracy requirements, and cryptographic limitations. At the layer level, protocols are designed in a modular manner to provide flexible support for various architectures. This SoK follows a *top-to-bottom* paradigm, covering model structures, threat models, cryptographic techniques, optimization methods, and experimental analysis. Rather than serving as a purely literature-driven survey, we emphasize analysis from a low-level implementation perspective. As shown in Figure 2, we use a *heatmap* to visualize research focus and identify gaps: red regions indicate areas that have attracted greater research attention, while blue regions point to aspects that warrant further exploration. The figure further distinguishes solution-oriented summaries (upper left) from analysis-oriented studies (lower right). Overall, this SoK aims to provide new insights through systematic analysis rather than mere summarization.

4 HE-Based Framework

HE-based frameworks (e.g., THE-X [12]) use HE as the only cryptographic tool throughout the entire evaluation of privacy-preserving inference. We analyzed twelve HE-based methods, as shown in Table 7 in Appendix B. The client first encrypts the token x and sends it to the server. For linear layers with weights w, the server computes $[\![x]\!] \boxdot [\![w]\!]$ ($\boxdot$ denotes HE-multiplication) based on the encrypted input. Some mixed frameworks (see Section VI) also process linear layers in this way (thus, the content of BOLT [33] is deferred to Section 6). Below, we discuss THE-X [12], PolyTransformer [54], Power-Softmax [55], FHE-BERT-Tiny [56], EncryptedLLM [57], Tricycle [58], NEXUS [17], THOR [59], Powerformer [60], MOAI [61], Arion [62], Zhang et al. [63] and Euston [64]. (For HE compilers/optimizers, such as Garimella et al. [65], please refer to Section 8.)

4.1 Advances in HE-Based Framework

Homomorphic-Friendly Design of Non-Linear Operators. Non-linear operators in Transformer models, such as GeLU, SoftMax, and LayerNorm, constitute the primary obstacle for HE inference, as HE schemes natively support only additions and multiplications. To address this mismatch, prior work has mainly followed two design strategies: approximation-based methods, and re-architecture methods.

Approximation-based methods simplify or replace complex non-linear components to reduce homomorphic evaluation cost while preserving inference accuracy. For instance, THE-X [12] and Thor [59] replace LayerNorm and similar operations with learnable affine transforms or pre-computed statistics. These approaches usually avoid full model retraining, making them practical for deployment.

Architecture-restructuring methods fundamentally adjust the Transformer design so that all computations are polynomial, naturally compatible with FHE. For example, FHE-BERT-Tiny [56] achieves this via operator substitution and knowledge distillation, while Donghwan et al. [66] and Powerformer [60] redesign the architecture. These methods ensure plaintext–ciphertext consistency during training but require retraining and careful trade-offs among expressiveness, training cost, and homomorphic efficiency.

Ciphertext Matrix Operations and Attention Acceleration. In transformer inference, the attention mechanism involves large-scale matrix multiplications, such as QK^{T} and attention-weighted summations, which dominate the runtime under HE. Since ciphertext rotation is one of the most expensive primitives in HE, reducing the number of rotations is a central optimization objective. NEXUS [17] and ENSI [67] propose SIMD packing techniques within a general system framework, where NEXUS reduces rotation overhead via a *binary-tree* based encoding layout, and ENSI achieves similar goals using a *Rotary Position Embedding* based encoding scheme. Tricycle [58] further improves batching efficiency through a *three-ring encoding* strategy, enabling more flexible ciphertext layouts for matrix multiplication to reduce rotation. MOAI [61] advances this line of work at the system level by proposing rotation-free activation evaluation methods, achieving orders-of-magnitude speedups for SoftMax and LayerNorm. Additionally, it introduces interleaved batching to further amortize rotation overhead. From an algorithmic perspective, Arion [62] proposes a *double Baby-Step Giant-Step (double-BSGS)* algorithm tailored for the three consecutive ciphertext matrix multiplications in attention. On BERT-Base, this approach reduces rotation counts by up to 82.5%, substantially improving linear-layer efficiency.

Multiplicative Depth Control and Bootstrapping Bottlenecks. In deep transformer models, the multiplicative depth of homomorphic computation grows rapidly, leading to severe noise accumulation and frequent invocation of bootstrapping. Multiple studies observe that bootstrapping often dominates end-to-end runtime. EncryptedLLM [57] provides a systematic evaluation of GPT-2 [68] under HE and shows that bootstrapping accounts for 50%–90% of inference latency. To address this bottleneck, the work introduces a GPU-accelerated CKKS HE library, offloading key operations such as bootstrapping and rotations to GPUs. Compared to CPU-based baselines, this approach achieves over 200× speedup, highlighting the significant potential of hardware acceleration for HE inference. Complementarily, Euston [64] shows that by adopting *binary exponentiation* for nonlinear approximations, the multiplicative depth can be reduced, enabling significant system-level speedups even on CPU platforms.

Beyond Whole-Model Encryption: Model Partitioning and Parameter Protection. Rather than encrypting the entire model, recent work explores partitioned architectures that selectively protect sensitive components under HE. A representative example is the Open-LLM + Private-LoRA framework [63], where the base model runs locally on the client in plaintext, and only LoRA adapters fine-tuned on private data are hosted on the server under HE. During inference, homomorphic computation is applied solely to these adapters, while the remaining layers execute on the client. This approach confines HE operations to a small subset of parameters, reducing encrypted computation and end-to-end latency, while providing a practical way to protect proprietary parameters without the overhead of whole-model encryption.

4.2 Research Directions

Early outsourcing paradigms relied on client participation for specific computations, particularly for non-linear functions that could not be efficiently evaluated under homomorphic encryption (HE) (e.g., ReLU or max operations). These operations were offloaded back to the client for decryption and subsequent re-encryption [12].

Open Problem 1. *Managing Multiplicative Depth and Bootstrapping Bottlenecks.* Deep models rapidly accumulate multiplicative depth during homomorphic evaluation, causing ciphertext noise growth and frequent expensive bootstrapping. Developing systematic strategies to manage the homomorphic computation chain, minimize bootstrapping, and support full-model inference remains an open problem.

Recent advances instead favor single-server, fully non-interactive inference within the HE framework, *without imposing any non-collusion assumptions among the involved parties*. In this configuration, the client submits encrypted inputs only once and can subsequently go offline. The server then performs the entirety of the inference procedure over ciphertexts, encompassing linear layers, attention mechanisms, and non-linear operators. By leveraging polynomial and piecewise function approximations, together with homomorphic sign functions to determine the relevant computation intervals, these protocols attain both correctness and privacy while substantially enhancing practical deployability.

Open Problem 2. *Security and Generalization of Model Partitioning and Minimal Encryption Paradigms.* Architectures like Open-LLM + Private-LoRA reduce homomorphic computation and improve performance, but their security boundaries, applicability, and task generalization remain to be fully evaluated.

Challenge 1. *Efficient Encoding of High-Dimensional Data and Control of Rotation Overhead.* Encoding schemes (e.g., SIMD packing, coefficient encoding, and nested encoding) map high-dimensional data into one-dimensional ciphertext slots. Efficiently performing this mapping while minimizing costly rotations remains a key challenge for linear operators.

Challenge 2. *Trade-offs in Non-Linear Layers.* Reducing latency without losing accuracy is challenging, as low-degree polynomial approximations reduce precision, while higher-degree or complex methods increase computation depth.

5 MPC-Based Framework

A large body of work leverages multiple servers under a non-collusion assumption to perform MPC. In this context, MPC-based frameworks are defined as those that rely exclusively on ASS, RSS, FSS, GC, OT, and related techniques that necessitate interactive protocols among multiple parties. We surveyed MPC-based approaches, which are summarized in Table 6 in Appendix B. Within such frameworks, the implementation of linear operations is largely standard: the majority of systems employ Beaver-triple–based secure multiplication, while a subset integrates more recent optimizations, such as the RSS-based multiplication protocol proposed in ASTRA [41]. Consequently, most prior work has focused on optimizing non-linear computations, which are widely recognized as the primary performance bottleneck in MPC-based Transformer inference.

Section 5.1 discusses key issues addressed by MPC-based frameworks, which also arise in other MPC-based tasks such as PPML, while Section 5.2 focuses on research directions that are specific to transformer-based models.

5.1 Advances in MPC-Based Framework

Secure and Efficient Evaluation of Non-linear Functions. Non-linear functions are widely recognized as the primary computational bottleneck in MPC-based transformer inference. Consequently, much work, such as SecureTLM [69], SecBERT [70], PrivTF [71], and SecureGPT [72], focuses on designing secure and efficient protocols for evaluating activation functions such as GeLU/SiLU and components like SoftMax and LayerNorm. For GeLU and SiLU, a dominant approach is piecewise or high-degree polynomial approximation, trading numerical accuracy for lower cryptographic cost. Examples include PUMA [14], which uses piecewise polynomials with comparison-based interval selection, and Truncformer [73] and SecFormer [74], which optimize polynomial degree and truncation strategies. Alternative numerical methods have also been explored: SHAFT [75] proposes an ODE-based protocol for accurate GeLU with fewer rounds, while Curl [76] uses discrete wavelet transform and optimized lookup tables to avoid expensive comparisons. For malicious security, Mosformer [77] introduces a LUT-based protocol with verifiable distributed point functions (VDPF) to evaluate GeLU over a reduced input domain.

SoftMax poses a significant challenge in secure computation due to its reliance on exponentiation, division, and normalization. To tackle this, existing research has converged on three primary strategies: *approximation-based protocols* that securely compute an approximate version of SoftMax (e.g., RNA-ViT [78], PUMA [14], SecFormer [74]); *MPC-friendly replacements* that substitute SoftMax with more efficient, cryptographically amenable operations, such as the ReLU-based attention in Privformer [13] and the linear attention in MLFormer [79]; and *model-aware co-design* , which jointly optimizes the model architecture and the secure protocol to enable learnable approximations of SoftMax, as exemplified by SAL-ViT [80] and MPCViT [81].

Private division, inverse square root, and exponential functions are the main sources of inefficiency for non-linear operators in MPC-based Transformer inference. They are commonly implemented via iterative numerical methods (e.g., Goldschmidt or Newton iterations and limit-based expansions), primarily because secret sharing natively supports only addition and multiplication, which involve repeated secure

multiplications and truncations, resulting in high communication overhead.

Communication Overhead Optimization. Minimizing communication is critical for practical MPC-based inference, especially over WAN. The overhead stems from two primary sources: the security model's verification requirements and the computational paradigm's interaction patterns [82]. *Security-Communication Trade-off:* Achieving malicious security necessitates verifiable operations (using MACs, commitments, etc.), which increase round complexity and byte count compared to semi-honest protocols. Recent work like Mosformer [77] addresses this by designing verifiable distributed point functions (VDPF) that enable constant-round comparisons and lookup table accesses, significantly cutting WAN latency. *Paradigm-Communication Trade-off:* Traditional ASS paradigms (Privformer [13], Comet [83]) incur per-layer online communication. In contrast, the FSS paradigm (exemplified by SIGMA [50]) moves most computation offline, slashing online traffic but introducing a trusted key generator. A spectrum of protocol innovations bridges this efficiency gap: these include constant-round comparisons, compressed nonlinear approximations (e.g., the discrete wavelet transform-compressed lookup tables in Curl [76]), domain reduction, sparse computations, and dynamic ciphertext adaptations (e.g., modulus switching in Mosformer [77]). Collectively, these techniques aim to reduce communication volume and rounds without compromising model accuracy.

Bitwidth-Aware Optimization for MPC-Based Private Inference. Optimizing MPC-based inference requires attention not only to communication overhead, but also to the fundamental cost of each cryptographic operation. This cost is largely dictated by the numerical representation of secret-shared values. Fixed-point arithmetic over the ring $\mathbb{Z}_{2^\ell}$, where ℓ denotes total bitwidth and f fractional precision, introduces a critical trade-off between computational efficiency and accuracy. While early frameworks often adopt a uniform 64-bit ring to reduce overflow and truncation risks, this conservative choice incurs substantial computation and communication overhead. Studies [73] show that bitwidth and precision directly affect model accuracy through error accumulation and probabilistic truncation, which can introduce non-negligible random errors in large models. However, through comprehensive analysis and reproduction experiments, we found that most works used a 37-bit ring with 12-bit fractional precision [33, 72, 84], which already offers sufficient accuracy. Therefore, employing a suitably flexible bitwidth does not affect the accuracy of the model (If the truncation protocol is precise enough).

Recent work, MosFormer [77], provides further empirical support for this viewpoint: distinct neural network operators exhibit heterogeneous sensitivity to numerical precision. Specifically, their analysis indicates that linear layers generally require higher precision to avoid numerical overflow, whereas nonlinear layers can reliably operate at reduced bitwidth. These findings motivate the development of operator-aware mixed-precision schemes. In comparison to uniformly high-precision configurations, such mixed-precision strategies yield superior computational efficiency.

Offline/Online Computation Paradigm. Beyond protocol design and bitwidth-level optimizations that reduce communication overhead, many latency-oriented optimizations are enabled by a strategic separation between offline preprocessing and online execution. This paradigm shifts the bulk of cryptographic setup and material generation to an offline phase, thereby minimizing latency-sensitive online interaction. *The Security-Driven Spectrum:* The offline/online split is largely determined by the security model. Under malicious security, where verifiability is crucial, frameworks like Mosformer [77] use a strict two-phase design: they precompute verifiable materials (e.g., authenticated Beaver triples, VDPF keys) offline to support efficient, constant-round online protocols. Under semi-honest security, the trade-offs favor flexibility. Systems like PUMA [14], SecFormer [74], and SHAFT [75] are online-centric, using lightweight primitives such as replicated secret sharing to reduce or hide offline costs and simplify deployment. *The Granularity Spectrum:* Offloading can occur at different granularities. Some frameworks (Comet [83], Fission [85]) selectively precompute specific expensive operations (e.g., random mask generation) while keeping most logic online, whereas others precompute entire cryptographic subroutines or data structures. The optimal granularity depends on the transformer's computation profile and available client/server storage.

5.2 Research Directions

The key technical issues discussed above are central to all MPC-based applications, including privacy-preserving machine learning and private set intersection. For private transformer-based model inference, additional directions arise, such as quantization (Section 7), but these still fundamentally rely on MPC. As summarized in Table 6 in Appendix B, existing work falls into several main trends.

End-to-End Inference Frameworks: Encrypted Runtime for Off-the-Shelf Transformers. Direct deployment of BERT [26], GPT [27], and other transformers in MPC is inefficient. The goal is to build a plug-and-play encrypted runtime achieving efficient inference with minimal model modification. i) *High-Precision Approximation* (PUMA [14], SHAFT [75], SecFormer [74]): Nonlinear functions such as SoftMax and GeLU are approximated using high-precision piecewise polynomials or lookup tables with optimized MPC protocols to maintain plaintext-level accuracy. ii) *Architecture Modification and Retraining* (MPCFormer [15], ML-Former [79]): Expensive nonlinear operations are simplified (e.g., quadratic functions, linear attention), and knowledge distillation or fine-tuning recovers accuracy. iii) *Low-Level Protocol and Operator Innovation* (Privformer [13], Mos-

former [77], Curl [76]): This improves the efficiency of fundamental cryptographic operators under Semi-honest or malicious security, including constant-round comparison, wavelet-based function evaluation, and secure truncation.

Component-Level Deep Optimizations: Targeting Performance Bottlenecks. Even within end-to-end frameworks, localized inefficiencies persist. Research addresses these through three main approaches: *Activation sparsity exploitation* (Comet [83]) securely bypasses computation and communication for zero-valued activations; *KV cache management* (MPCache [86]) introduces MPC-aware eviction policies to reduce overhead in autoregressive decoding, and designs a query-aware dynamic selection algorithm that discards unimportant KV cache entries and activates only a small subset for attention computation. And *computation graph optimization* (MERGE [87]) restructures operator sequences and reuses intermediate states to minimize redundant cryptographic operations. Together, these component-level optimizations demonstrate that the systematic analysis of the inference workflow can yield substantial efficiency improvements beyond protocol-level advancements.

Neural Architecture Design (NAS): MPC-Friendly Model Families. Conventional neural network architectures are not intrinsically designed or parameterized for efficient execution within MPC frameworks. Recent studies, including MPCViT [81], RNA-ViT [78], and SAL-ViT [80], leverage NAS in combination with advances in attention mechanisms (e.g., linearized SoftMax) to identify architectures that optimally balance MPC-induced latency against predictive performance. These research efforts yield classes of models that exhibit particularly strong suitability for deployment within MPC frameworks.

Computation Paradigm Innovations: Beyond Traditional MPC. To address the efficiency limits of conventional MPC, recent work explores alternative trust-computation models. The *hybrid paradigm*, exemplified by Fission [85], runs linear layers under MPC and nonlinear activations on obfuscated plaintext, yielding large speedups under relaxed trust. In contrast, the *FSS paradigm*, exemplified by SIGMA [50, 88], shifts the majority of computation to an offline phase through a trusted key generation procedure, thereby substantially reducing online communication overhead at the expense of stronger trust assumptions.

Open Problem 3. *High communication overhead limits the application scenarios of MPC-based frameworks.* MPC-based solutions typically require communication and interaction, and the latency of each round is affected by network bandwidth and delay, which ultimately leads to a significant drop in overall inference performance.

Open Problem 4. *Universality and Security of Truncation Protocols.* Truncation is a ubiquitous operation for fixed-point precision management, but is costly in practice. How to design efficient, general truncation protocols with strong security guarantees, especially under the malicious adversary model, remains open.

Challenge 3. *The efficient and accurate approximation of operations such as* $\frac{x}{y}$, $\frac{1}{\sqrt{y}}$, exp, tanh, *and comparison (cmp) over secret values x and y is a key challenge.* These non-linear functions must be realized indirectly via additional protocols, typically incurring high communication rounds, frequent truncation operations, and non-trivial trade-offs between precision and efficiency.

Insight 1. *Increasing importance of offline/online separation for low-latency and malicious-secure inference.* Future optimization directions may include: 1) further reducing the cost of offline precomputation under specific security models, and 2) designing systems that amortize offline computation across multiple inference requests.

Insight 2. *Towards TTP-Free and Integrated FSS.* FSS is a promising foundation for low-latency secure protocols. Removing its dependence on a trusted third party and integrating it more tightly with linear secret sharing schemes could greatly increase its practicality and impact, making this a key direction.

6 HE-MPC Hybrid Framework

Many oblivious inference frameworks use both HE and ASS (or GC) for the best of both worlds, e.g., reducing runtime. As a result, this line of work overlaps with both HE and MPC approaches.

6.1 Advances in Hybrid Framework

Protocol-Oriented Decomposition of Linear and Non-Linear Operators. In private inference tasks involving hybrid cryptographic techniques, a common paradigm is to employ HE for linear operations, such as matrix multiplication. In contrast, nonlinear operations, including SoftMax, LayerNorm, and GeLU, are handled by converting from HE to ASS and evaluated using MPC protocols. Representative works include Iron[89], which first formally established the paradigm of using HE for linear operators and MPC for non-linear operators. CipherGPT [90] extended this paradigm to autoregressive generative models. CipherFormer [91] adopts a hybrid BFV–GC approach. Primer [52] further introduced offline/online separation optimization based on this decomposition. The essential

difference from HE-based approaches lies in the fact that HE requires all operators to be transformed into HE-friendly forms, whereas hybrid methods focus on the rational design of protocol boundaries and interface management.

Hybrid-Friendly Linear Algebra: Packing, Fusion, and Rotation Reduction. This line of research partially overlaps with HE–based methods, for example, in reducing the number of ciphertexts and rotations through the design of matrix encodings. Existing works mainly explore SIMD encoding, coefficient encoding, and nested encoding schemes. SIMD encoding, widely adopted by works such as BOLT [33], preserves the consistency between input and output layouts, thereby enabling the consecutive evaluation of multiple linear operators at the cost of frequent and expensive ciphertext rotations. In contrast, Iron [89] builds on Cheetah's [92] coefficient encoding, mapping linear operators to polynomial multiplication and almost completely eliminating rotations. However, this breaks intermediate layout consistency and is therefore more suitable for single-layer computation or as an interface operator in HE–MPC hybrid execution. Nested encoding, as proposed by PrivCirNet [93], further combines the advantages of both approaches and achieves theoretically optimal complexity in parallel settings but relies on discrete Fourier transforms (DFT) and is tightly coupled with bootstrapping. Beyond encoding schemes, another research direction focuses on reducing rotation overhead at the algorithmic level, including hoisting, BSGS optimization, as well as column packing and multi-head packing in BOLT [33] and BLB [94]. These techniques share the common goal of reducing the most expensive operation in HE, namely, ciphertext rotation. For matrix multiplication, different works further explore trade-offs between rotation count and multiplicative depth in ct-pt and ct-ct settings: BOLT [33] achieves state-of-the-art complexity in the ct-pt case, while in the ct-ct setting, BOLT [33] and BLB [94] obtain better overall performance by reducing multiplicative depth.

Under the hybrid HE–MPC execution model, optimization of linear operators further extends to cross-operator and cross-layer collaboration. Since switching between HE and MPC inevitably introduces secure conversion and communication overhead, fusing multiple linear operators into larger HE computation blocks to reduce the number of HE-MPC transitions becomes a key factor in improving end-to-end performance. Nimbus and BLB [94] represent explorations along this direction. Nimbus combines client-side outer product with row-wise packing to alter the computation and communication paradigm, producing more compact ciphertext outputs. BLB [94] further breaks layer boundaries by systematically fusing consecutive linear operators at the operator granularity, merging previously scattered HE computations and conversions into a single HE block to enable system-level, cross-protocol optimization.

Secure Computation of Nonlinear Operators. Nonlinear functions pose challenges for efficient secure computation. Existing approaches focus on three main directions: i) precise computation using MPC or GC with LUTs, as in Iron [89], Primer [52], Liu et al. [95], and CipherGPT [90], which preserve accuracy but incur high communication overhead, ii) polynomial approximations, as in BOLT [33], FASTLMPI [53], BumbleBee [84], and EAST [96], which reduce computation by mapping nonlinear functions to low-degree polynomials, and iii) adaptive approximation methodologies, exemplified by CipherPrune [97], Nimbus [98], and PermLLM [99], which dynamically modify the protocol-level realizations of nonlinear operators as a function of input characteristics or batch organization.

6.2 Research Directions

HE–MPC hybrid protocols combine the low-communication advantage of HE with the high-precision benefit of MPC, and their development can be divided into three stages. These methods reflect a shift from passive computation to input-aware optimization in the encrypted domain.

Basic Hybrid Protocols. Iron [89] established a layer-wise hybrid paradigm, where linear layers (e.g., matrix multiplication, fully connected layers) are efficiently computed with HE to reduce communication, while nonlinear layers (e.g., SoftMax, GeLU, LayerNorm) are computed via MPC or Garbled Circuits to ensure accuracy.

Operator-Level Fusion and Fine-Grained Optimization. Nonlinear layers contain many internal linear operators, enabling cross-layer fusion and reducing protocol switching. BLB [94] proposed a fine-grained operator fusion method by merging multiple linear operators from LayerNorm, subsequent FC layers, and GeLU layers into a single HE block, which reduces HE–MPC conversions by over 50% and significantly lowers communication and latency. PrivCirNet [93] introduced a model–protocol co-design by transforming weight matrices into block-circulant matrices, allowing efficient HE convolution-based multiplication.

Complex Task and Input-Aware Optimization in the Encrypted Domain. The hybrid framework is evolving from static computation toward input-aware, encrypted-domain optimization. CipherPrune [97] dynamically prunes redundant tokens in the encrypted state and applies polynomial degree reduction to reduce the $O(n^2)$ complexity of attention mechanisms. PermLLM [99] explores selectively revealing a small number of low-risk parameters to improve efficiency. Notably, they adopt additive secret sharing on real numbers.

Open Problem 5. *The security of HE–MPC conversions is lacking.* BLB [94] provides a secure conversion protocol between CKKS and MPC. For schemes like BFV, which operate over a different domain than CKKS, secure conversion protocols and corresponding security analyses remain lacking.

Insight 3. *Advantages of Combining HE and MPC.* Integrating HE and MPC in a hybrid framework brings two key advantages. 1) Hybrid HE–MPC eliminates costly HE bootstrapping, 2). while using HE for linear computations, reducing overall communication and improving efficiency.

7 Model Optimization and Quantization

7.1 Optimization

Beyond cryptographic optimizations, model-level improvements have become a key direction for enhancing private inference. MPCFormer [15] replaces GeLU or SoftMax with quadratic functions or ReLU and distills intermediate representations to accelerate inference with minimal accuracy loss. Cryptpeft [100] replaces the SoftMax with LinAtten by using the Parameter-Efficient Fine-Tuning. PriViT [101] adaptively selects computation strategies for different ViT attention heads, while MPCViT [81] uses architecture search to balance accuracy and latency across heterogeneous attention spaces. Most prior optimizations focus on accelerating cryptographic execution of existing models (Song et al. [102]), whereas Aero [103] shows that modifying model structures without understanding their impact on internal information flow can hit fundamental performance collapse limits, and thus proposes an entropy-guided framework for system, model, and cryptography co-optimization. MARILL [104] proposes a layer-freezing strategy that partitions the model into public and private components, applying an MPC engine only to the private part for secure computation. The model architecture is adapted through an MPC-minimization–oriented fine-tuning framework. However, this approach suffers from the same issue as the strategy proposed by Zhang et al. [63]: is the boundary between public and private components truly fixed? If not, it may introduce potential privacy leakage risks.

7.2 Quantization

Quantization is an efficient approach for private inference and has been explored at multiple levels. Lu et al. [36] propose a native 8-bit secure inference paradigm using integer operations and table lookups, greatly reducing communication. Ditto [16] combines MPC-friendly hierarchical quantization with knowledge distillation, using mixed bit-widths across layers and efficient type-conversion for secure computation. FastQuery [105] tackles LLM embedding lookup bottlenecks via per-channel mixed-precision quantization and one-hot-aware data packing, drastically lowering communication. Ditto [16] extends this with static quantization and quantization-aware distillation, converting floating-point numbers to low-bit fixed-point representations and using mixed bit-widths for efficiency.

8 Practical Deployment Issues

8.1 HE Frameworks Deployment

HE parameters. To mitigate numerical overflow in consecutive ciphertext multiplications, HE inference frameworks based on BFV or BGV (e.g., BOLT [33], Iron [89] , and BumbleBee [84]) typically adopt a larger plaintext modulus. However, since BFV does not support explicit scale compression, the numerical scale of operands grows exponentially across successive multiplications. In contrast, CKKS and its variants (e.g., RNS-CKKS) employ rescaling after each multiplication to actively discard the least significant bits and compress the numerical scale back to a controllable range, thereby keeping the growth of ciphertext bit-width approximately linear. As a result, CKKS has been widely adopted by recent systems such as NEXUS [17], Euston [64], MOAI [61], and Arion [62].

Developer-Friendly Compilers. The objective is to abstract away cryptographic complexity from non-expert users while supporting efficient execution of privacy-preserving inference tasks. i). They assist HE parameter selection, which is crucial for performance: CHET [106] and SEALion [107] derive parameters from the dataset and computation graph, while nGraph-HE2 [108] favors small moduli (e.g., 32-bit) to reduce CPU overhead and automatically selects safe scaling factors. ii). Compilers perform automatic packing and data-layout optimization to exploit SIMD in RLWE-based HE. Early systems such as CHET [106], HEIR [109], and EVA [110] use expert-defined fixed layouts, whereas later work (AHEC [111], Porcupine [112], Coyote [113]) moves toward automated and flexible vectorization, with richer layout abstractions in HELayer [114], Fhelipe [115], HECO [116], and FlexHE [117]. iii). For CKKS-based inference, compilers jointly optimize scale management and bootstrapping placement: EVA [110], HECATE [118], and ELASM [119] focus on scale control, while Fhelipe [115], DaCapo [120], ReSBM [121], HALO [122], ANT-ACE [123], and Orion [124] further enable automatic bootstrapping under depth and latency constraints.

Challenge 4. *Compiler-Hardware Co-Optimization for HE.* Developing a compiler for private transformer-based model inference requires deep expertise in system architecture, compiler optimization, and cryptography, making it a highly non-trivial task.

Insight 4. *Unified Computation Optimization for Private Transformer.* Treating private inference tasks as a unified computation task, rather than optimizing individual operators or protocols, enables compiler-level global optimization, moving beyond per-protocol or per-operator approaches.

Insight 5. *Compiler-Hardware Co-Optimization for HE.* Compiler-hardware co-optimization for HE leverages real-valued schemes like CKKS, avoids MPC-style interaction, and enables GPU acceleration, speeding up privacy-preserving transformer inference.

8.2 HE-MPC Frameworks Deployment

Mixed-Precision Computation. In MPC-based private inference, communication often dominates and scales with data bit-width. Traditional approaches use uniform high-precision representations (e.g., 64-bit integer rings with 16-bit fractional precision) to ensure correctness, leading to excessive overhead for nonlinear operations. Linear layers like matrix multiplication require large rings to avoid overflow, while nonlinear activations (ReLU, GeLU) and SoftMax have bounded inputs and can be computed in smaller rings with lower precision, reducing communication. Representative contributions in this area encompass Mosformer's [77] mixed-modulus switching scheme and Ditto's [16] static, layer-wise quantization framework with mixed-precision execution.

MPC Multi-Round Interaction and Network Adaptivity. The multi-round interaction inherent in MPC protocols makes them highly sensitive to network latency. In high-latency WAN environments, both communication volume and interaction rounds can significantly increase online inference latency. One line of work mitigates this issue by reducing the number of communication rounds, for example, through constant-round SoftMax protocols such as those proposed in SHAFT [75]. This phenomenon appears in both MPC-based and HE–MPC hybrid frameworks: while these approaches achieve sufficiently high efficiency in LAN settings, their performance can degrade in WAN environments, where they may even be outperformed by HE-based solutions such as NEXUS [17].

Insight 6. *Communication–Computation Trade-offs under Network-Aware Private Inference.* When network latency is sufficiently low, additional communication can be acceptable, or even beneficial for reducing overall latency by alleviating computational load or protocol depth. Therefore, private inference should adopt a unified, network-aware optimization perspective, balancing communication and computation jointly rather than optimizing either in isolation.

Open Problem 6. *Deployment Constraints of MPC-based Inference.* How to systematically reduce the communication cost and interaction rounds of MPC, while preserving security and accuracy, remains an open problem.

Table 3: Statistics on extensibility of PPML frameworks to large model private inference.

Extensibility	PPML Works
×	[41], [125], [126], [127], [128], [129], [10], [130] [131], [132], [133], [134], [135], [136], [137], [138] [139], [140], [141], [142], [143], [106], [144], [108] [145], [146], [147], [11], [148], [149], [150], [92] [151], [152], [153], [154], [155], [156], [9], [157] [158], [159], [160], [161], [162], [163]
✓	[164], [165], [166],[167], [168], [169]

The left column indicates whether the technique can be extended (✓) or not (×) to transformer-based large models; the right column lists representative PPML works.

8.3 Extensibility of The Traditional PPML

Existing privacy-preserving inference frameworks, such as ABY [137, 164], ABY^3 [139], EzPC [168], CrypTen [166], CrypTFlow2 [165], SecretFlow [169], and MP-SPDZ [160], are designed for generality and can support a variety of traditional machine learning models, such as ResNet. However, Lu et al. in BumbleBee [84] note that these frameworks are difficult to directly scale to transformer-based model inference. Large models like BERT [26] involve much larger matrix operations and more complex nonlinear functions (e.g., GeLU, SoftMax), whereas traditional models mainly use simpler nonlinearities like ReLU. We surveyed 54 prior works and found that only 6 frameworks can be directly applied to large model privacy inference (see Table 4), based on whether their protocols support the full range of operators required by these models [18].

Insight 7. *Hybrid HE Compilation Beyond a Single Scheme.* Is it possible to build an end-to-end compiler, akin to HEIR [109], that integrates multiple HE schemes, mapping linear layers to CKKS-style arithmetic and nonlinear operations to TFHE Boolean circuits, with large-circuit optimizations (e.g., AutoHoG [170]), to enable efficient, unified privacy-preserving inference?

9 Network-Aware Performance

We examined the openness of all surveyed works, covering model choices, communication costs, inference latency, code availability (see Appendix B), and GPU support. In our experimental evaluation, Figures 3, 4, and 5 focus on baseline performance comparisons. Using a unified BERT-Base architecture, this test evaluates performance under standard network profiles to investigate the impact of network environments. And the data uses random data of the same scale as BERT-Base, which does not affect the experimental conclusions. We observe that network conditions, especially latency and bandwidth, have a significant impact on system performance, motivating additional evaluations under WAN settings with

varying bandwidths to assess robustness in non-ideal network environments. For the evaluations in Tables 9, 10, 11, and 12 in Appendix B, since the architectural designs of the various baseline frameworks differ significantly, making a completely unified reproduction highly challenging, we re-implemented these methods by strictly adhering to the specific model and network configurations from their original papers to guarantee evaluation fidelity. And the data strictly follows that provided in the original papers; for data not provided, we have marked it in the experimental tables. For a more comprehensive description of the detailed architectural parameters and environment setups, please refer to Appendix B. Notably, several open-source codes could not be executed successfully, so we use the performance results reported in their original papers. Even according to their own reported results, these systems still fall far short of meeting real-time interactive requirements. We also summarize the commonly used cryptographic backbones in Table 5 in Appendix B.

Notably, several open-source codes could not be executed successfully, so we use the performance results reported in their original papers. Even according to their own reported results, these systems still fall far short of meeting real-time interactive requirements. We also summarize the commonly used cryptographic backbones in Table 5 in Appendix B.

Inference Latency. Figure 3 shows the relationship between accuracy and inference latency for representative methods selected from HE-based, MPC-based, and hybrid approaches on the BERT-Base model in a LAN setting. SIGMA [50] achieves the lowest inference latency by relying on a trusted third party and offloading most computations to the offline phase, benefiting from GPU-accelerated FSS. We use different colors to distinguish techniques: MPC-based methods perform best in LAN environments, hybrid approaches lie in between, while HE-based methods exhibit higher latency overall. In terms of accuracy, private Transformer inference is no longer a major bottleneck, as most frameworks achieve 66–70% accuracy on RTE, close to plaintext performance.

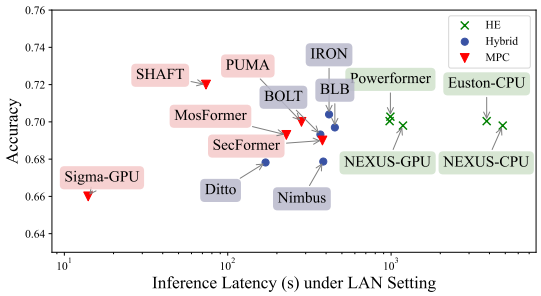


Figure 3: Relationship between inference accuracy and inference latency of BERT-Base in a LAN environment. The system settings are described Appendix B.

Re-evaluation over WAN. As shown in Figures 4 and 5 , it can be observed that as network bandwidth decreases, the inference efficiency of MPC-based solutions decreases significantly. Under WAN conditions, MPC-based frameworks appear on the far right of the figure with the highest latency. Most hybrid methods remain in the middle, indicating strong adaptability to network variations.

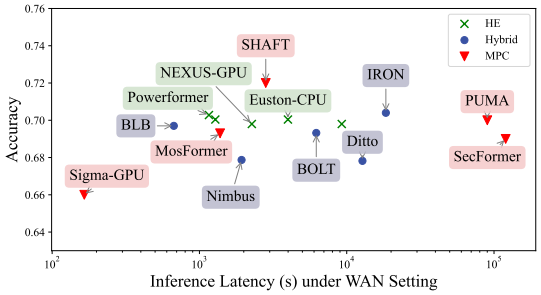


Figure 4: Relationship between inference accuracy and inference latency of BERT-Base in a WAN environment.

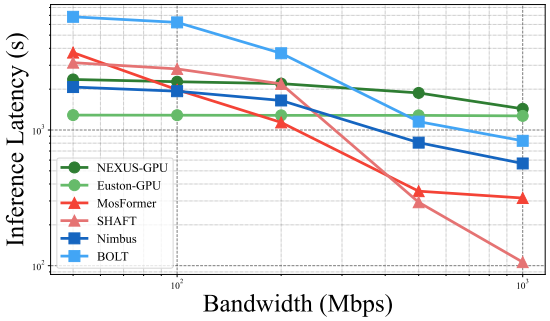


Figure 5: Relationship between inference latency and network settings on BERT-Base.

GPU Trumps. As shown in Figures 3 and 4, GPUs play a crucial role. Sigma, EUSTON [64], and NEXUS [17] all utilize GPUs. We expect future frameworks to continue leveraging GPUs even more extensively. Nevertheless, studies show that FPGA/ASIC accelerators outperform GPUs in performance and energy efficiency for core FHE operations (NTT, key switching, bootstrapping) [123]. Thus, a promising direction is developing heterogeneous architectures integrating specialized FPGA/ASIC accelerators for efficient FHE execution.

10 Discussion & Conclusion

In the following discussion, we examine this goal of achieving efficient private Transformer-based model inference from three perspectives: fairness of comparisons, the gap between protocol design and practical deployment, and the conclusion.

Comparison. A fair comparison across privacy-preserving inference systems requires attention to several key factors. i) Number of parties. Latency is affected by the number

of participating parties. Even under the same protocol, SecureGPT [72] shows that increasing the number of parties can reduce latency due to communication overhead. Moreover, existing works often adopt different protocols across 2PC and 3PC settings: 2PC multiplication typically relies on OT or HE-based Beaver triples, which are generally less efficient than replicated secret sharing used in 3PC, making direct comparisons non-trivial. ii) Frameworks with explicit offline/online phases are not directly comparable to those without. For example, Primer reports 399.4 s offline preprocessing and 35.4 s online latency for BERT-Base, whereas BLB [94] without offline preprocessing requires 452.60 s under the same model setting. iii) Model-level optimizations. Performance also depends on whether model-specific optimizations are applied. For instance, BOLT achieves 91 s inference latency with word elimination, compared to 185 s without it on BERT-Base.

Research Gap & Guidance. Despite substantial progress, several important gaps remain. i) Most existing works focus on protocol-level optimizations or model compression. Future research could elevate optimization to the computation-graph level [108], enabling protocol-aware graph transformations and operator scheduling tailored to underlying cryptographic primitives. ii) Mixed-precision inference remains underexplored. Recent efforts, such as Mosformer [77], demonstrate that precision heterogeneity can significantly reduce communication cost, suggesting a promising direction, particularly for MPC-based systems that are highly sensitive to network conditions. iii) Support for malicious adversary models is limited. Only a small number of works address malicious security, yet stronger threat models are essential for deploying large models in high-stakes privacy-sensitive scenarios. iv) Verifying the correctness of returned inference results while preserving user privacy remains challenging. Initial attempts, such as ZK-GPT [171], suggest that verifiable private inference is a meaningful yet largely open problem. v) Most existing privacy-preserving inference work focuses on relatively small models such as BERT (4M–340M parameters). Even for larger models like LLaMA-7B (7B parameters), frameworks such as PUMA [14] require five minutes for inference, which is far from real-time deployment. Future research should address the key direction: scaling to larger models and developing techniques that support real-time inference on commonly used large models beyond LLaMA-7B.

Figure 5 illustrates this phenomenon under LAN and WAN settings and motivates careful selection of privacy-preserving technologies according to application scenarios and latency requirements. Accordingly, i) High-bandwidth, low-latency scenarios: MPC-based frameworks are preferable, show high performance variability under high bandwidth, and latency can be further reduced by slightly increasing communication. ii) Low-bandwidth, high-latency scenarios: HE-based frameworks show high performance variability under low bandwidth, especially when GPU acceleration is available. iii) Hybrid frameworks: Their flexibility allows dynamic adjustment of HE–MPC usage to accommodate a wider range of conditions. If the network conditions vary in real time, hybrid frameworks are a good choice.

Our Conclusion. This paper systematically reviews private transformer-based model inference techniques, covering HE-based, MPC-based, and hybrid HE/MPC designs, as well as model-level optimizations. Based on our analysis of existing work, a unifying goal emerges: *real-time interactive inference*, i.e., minimizing inference latency and narrowing the gap to plaintext execution. A key conclusion is that no single approach dominates; each occupies a distinct region of the design space with trade-offs in performance, deployability, and privacy guarantees. Hybrid designs demonstrate that combining paradigms can improve efficiency and enhance adaptability to different deployment scenarios.

Ethical Considerations

This work presents a systematization of knowledge on privacy-preserving inference for transformer-based models, providing a detailed analysis, insights into the evolution of the field, and perspectives on future directions. We further assess the reproducibility of prior work by re-evaluating all studies. The literature analyzed in this SoK is explicitly designed to protect the privacy of input data and model parameters, thereby adhering to ethical standards. Overall, this work contributes to the development of ethical and responsible privacy-preserving inference for large language models. However, the deployment of PETs introduces dual-use risks for stakeholders like regulators and affected individuals. To this end, robust security auditing mechanisms are indispensable. Recent work has already explored feasible approaches to auditing encrypted execution without revealing sensitive information. Consequently, such dual-use risks can be reasonably mitigated.

Open Science

We are committed to promoting transparency, reproducibility, and open access in scientific research. As part of our Open Science policy, this work takes the form of a SoK study, in which we systematically analyze existing works. We have provided, via an anonymous link, the URLs and code links of all surveyed private transformer-based model inference studies to facilitate peer review and community evaluation. We recognize the importance of open access to research resources and encourage others to build upon this work.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need.

Advances in neural information processing systems, 30, 2017.

[2] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.

[3] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5625–5644, 2024.

[4] Zheyi Chen, Liuchang Xu, Hongting Zheng, Luyao Chen, Amr Tolba, Liang Zhao, Keping Yu, and Hailin Feng. Evolution and prospects of foundation models: From large language models to large multimodal models. *Computers, Materials & Continua*, 80(2), 2024.

[5] Biwei Yan, Kun Li, Minghui Xu, Yueyan Dong, Yue Zhang, Zhaochun Ren, and Xiuzhen Cheng. On protecting the data privacy of large language models (llms): A survey. *arXiv preprint arXiv:2403.05156*, 2024.

[6] Zi Liang, Pinghui Wang, Ruofei Zhang, Nuo Xu, Shuo Zhang, Lifeng Xing, Haitao Bai, and Ziyang Zhou. Merge: Fast private text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19884–19892, 2024.

[7] Prajwal Panzade, Daniel Takabi, and Zhipeng Cai. Medblindtuner: Towards privacy-preserving fine-tuning on biomedical images with transformers and fully homomorphic encryption. In *AI for health Equity and Fairness*, pages 197–208. Springer, 2024.

[8] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pages 896–902. IEEE, 2015.

[9] Wenxuan Zeng, Tianshi Xu, Yi Chen, Yifan Zhou, Mingzhe Zhang, Jin Tan, Cheng Hong, and Meng Li. Towards efficient privacy-preserving machine learning: A systematic review from protocol, model, and system perspectives. *ArXiv*, abs/2507.14519, 2025.

[10] Lucien K. L. Ng and Sherman S. M. Chow. Sok: Cryptographic neural-network computation. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 497–514, 2023.

[11] Congcong Chen, Lifei Wei, Jintao Xie, and Yang Shi. Privacy-preserving machine learning based on cryptography: A survey. *ACM Transactions on Knowledge Discovery from Data*, 19:1 – 33, 2025.

[12] Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. THE-X: Privacy-preserving transformer inference with homomorphic encryption. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3510–3520, Dublin, Ireland, May 2022. Association for Computational Linguistics.

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

[14] Dong, Ye, Lu, Wen-Jie, Zheng, Yancheng, Wu, Haoqi, Zhao, Derun, Tan, Jin, Huang, Zhicong, Hong, Cheng, Wei, Tao, Chen, Wen-Guang, and Zhou, Jianying. Puma: Secure inference of llama-7b in five minutes. *Security and Safety*, 4:2025014, 2025.

[15] Dacheng Li, Hongyi Wang, Rulin Shao, Han Guo, Eric Xing, and Hao Zhang. Mpcformer: Fast, performant, and private transformer inference with mpc. In *The Eleventh International Conference on Learning Representations*, 2023.

[16] Haoqi Wu, Wenjing Fang, Yancheng Zheng, Junming Ma, Jin Tan, Yinggui Wang, and Lei Wang. Ditto: Quantization-aware secure inference of transformers upon mpc. *ArXiv*, abs/2405.05525, 2024.

[17] Jiawen Zhang, Jian Liu, Xinpeng Yang, Yinghao Wang, Kejia Chen, Xiaoyang Hou, Kui Ren, and Xiaohu Yang. Secure transformer inference made non-interactive. In *NDSS*, 2025.

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

[19] Martin Zbudila, Ajith Suresh, Hossein Yalame, Omid Mirzamohammadi, Aysajan Abidin, and Bart Preneel. Sok: Connecting the dots in privacy-preserving ml-systematization of mpc protocols and conversions between secret sharing schemes. *Cryptology ePrint Archive*, 2025.

[20] Jinglong Luo, Yehong Zhang, Zhuo Zhang, Shiyu Liu, Ye Dong, Haoran Li, Yue Yu, Hui Wang, Xun Zhou, and Zenglin Xu. Cryptography-based privacy-preserving large language models: a lifecycle survey of frameworks, methods, and future directions. *Artificial Intelligence Review*, 2025.

[21] Davide Andreoletti, Alessandro Rudi, Emanuele Carpanzano, and Tiziano Leidi. Privacy-preserving llm inference in practice: A comparative survey of techniques, trade-offs, and deployability. *Cryptology ePrint Archive*, 2026.

[22] David Cerdeira, José Martins, Nuno Santos, and Sandro Pinto. {ReZone}: Disarming {TrustZone} with {TEE} privilege reduction. In *USENIX Security*, pages 2261–2279, 2022.

[23] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *IEEE symposium on security and privacy (SP)*, pages 1897–1914. IEEE, 2022.

[24] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[25] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. A survey on federated learning: challenges and applications. *International journal of machine learning and cybernetics*, 14(2):513–535, 2023.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[27] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, and et al. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

[28] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, and et al. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023.

[29] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *Proceedings of the 22nd ACM international conference on Multimedia*, 2014.

[30] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Z. Chen, Craig Citro, and et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *ArXiv*, abs/1603.04467, 2016.

[31] Shashank Mohan Jain. Hugging face. In *Introduction to transformers for NLP: With the hugging face library and models to solve problems*, pages 51–67. Springer, 2022.

[32] Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. Cogltx: Applying bert to long texts. *Advances in Neural Information Processing Systems*, 33:12792–12804, 2020.

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

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

[35] Jean-Luc Watson, Sameer Wagh, and Raluca Ada Popa. Piranha: A {GPU} platform for secure computation. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 827–844, 2022.

[36] Tianpei Lu, Bingsheng Zhang, Xiaoyuan Zhang, and Kui Ren. A new ppml paradigm for quantized models. In *NDSS*, 2025.

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

[38] Amos Beimel and Benny Chor. Secret sharing with public reconstruction. In *Annual International Cryptology Conference*, pages 353–366. Springer, 1995.

[39] Daoshun Wang, Lei Zhang, Ning Ma, and Xiaobo Li. Two secret sharing schemes based on boolean operations. *Pattern Recognition*, 40(10):2776–2785, 2007.

[40] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 337–367. Springer, 2015.

[41] Harsh Chaudhari, Ashish Choudhury, Arpita Patra, and Ajith Suresh. Astra: High throughput 3pc over rings with application to secure prediction. *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, 2019.

[42] Andrew C Yao. Circuits and local computation. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 186–196, 1989.

[43] Vijay Kumar Yadav, Nitish Andola, Shekhar Verma, and S Venkatesan. A survey of oblivious transfer protocol. *ACM Computing Surveys (CSUR)*, 54(10s):1–37, 2022.

[44] Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated ot

with small communication. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1607–1626, 2020.

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

[46] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual cryptology conference*, pages 868–886. Springer, 2012.

[47] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

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

[49] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[50] Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. Sigma: Secure gpt inference with function secret sharing. *Proc. Priv. Enhancing Technol.*, 2024:61–79, 2024.

[51] Christopher Harth-Kitzerow, Ajith Suresh, and Georg Carle. Sok: Truncation untangled: Scaling fixed-point arithmetic for privacy-preserving machine learning to large models and datasets. *Proceedings on Privacy Enhancing Technologies*, 2025.

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

[53] Yuntian Chen, Zhanyong Tang, Tianpei Lu, Bingsheng Zhang, Zhiying Shi, and Zheng Wang. Accelerating private large transformers inference through fine-grained collaborative computation. *IEEE Transactions on Information Forensics and Security*, 20:7482–7497, 2025.

[54] Itamar Zimerman, Moran Baruch, Nir Drucker, Gilad Ezov, Omri Soceanu, and Lior Wolf. Converting transformers to polynomial form for secure inference over homomorphic encryption. *ArXiv*, abs/2311.08610, 2023.

[55] Itamar Zimerman, Allon Adir, Ehud Aharoni, Matan Avitan, Moran Baruch, Nir Drucker, Jenny Lerner, Ramy Masalha, Reut Meiri, and Omri Soceanu. Power-softmax: Towards secure llm inference over encrypted data, 2024.

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

[57] Leo de Castro, Daniel Escudero, Adya Agrawal, Antigoni Polychroniadou, and Manuela Veloso. EncryptedLLM: Privacy-preserving large language model inference via GPU-accelerated fully homomorphic encryption. In *Forty-second International Conference on Machine Learning*, 2025.

[58] Lawrence Lim, Vikas Kalagi, Divyakant Agrawal, and Amr El Abbadi. Tricycle: Private transformer inference with tricyclic encodings. *Cryptology ePrint Archive*, 2025.

[59] Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. Thor: Secure transformer inference with homomorphic encryption. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 3765–3779, 2025.

[60] Dongjin Park, Eunsang Lee, and Joon-Woo Lee. Powerformer: Efficient and high-accuracy privacy-preserving language model with homomorphic encryption. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11090–11111, Vienna, Austria, July 2025. Association for Computational Linguistics.

[61] Linru Zhang, Xiangning Wang, Jun Jie Sim, Zhicong Huang, Jiahao Zhong, Huaxiong Wang, Pu Duan, and Kwok Yan Lam. Moai: Module-optimizing architecture for non-interactive secure transformer inference. *Cryptology ePrint Archive*, 2025.

[62] Linhan Yang, Jingwei Chen, Wangchen Dai, Shuai Wang, Wenyuan Wu, and Yong Feng. Arion: Attention-optimized transformer inference on encrypted data. *Cryptology ePrint Archive*, 2025.

[63] Ruoyan Zhang, Zhongxiang Zheng, and Wankang Bao. Practical secure inference algorithm for a fine-tuned large language model based on fully homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 21:17–29, 2026.

[64] Xinwen Gao, Shaojing Fu, Lin Liu, Zhuotao Liu, Yuchuan Luo, and Yongjun Wang. Euston: Efficient and user-friendly secure transformer inference with non-interactivity. *Cryptology ePrint Archive*, 2026.

[65] Karthik Garimella, Negar Neda, Austin Ebel, Nandan Kumar Jha, and Brandon Reagen. Network and compiler optimizations for efficient linear algebra kernels in private transformer inference. *arXiv preprint arXiv:2512.11135*, 2025.

[66] Donghwan Rho, Taeseong Kim, Minje Park, Jung Woo Kim, Hyunsik Chae, Ernest K. Ryu, and Jung Hee Cheon. Encryption-friendly llm architecture, 2025.

[67] Zhiyu He, Maojiang Wang, Xinwen Gao, Yuchuan Luo, Lin Liu, and Shaojing Fu. Ensi: Efficient non-interactive secure inference for large language models. *arXiv preprint arXiv:2509.09424*, 2025.

[68] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[69] Yuntian Chen, Xianjia Meng, Zhiying Shi, Zhiyuan Ning, and Jingzhi Lin. Securetlm: Private inference for transformer-based large model with mpc. *Information Sciences*, 667:120429, 2024.

[70] Hai Huang and Yongjian Wang. Secbert: Privacy-preserving pre-training based neural network inference system. *Neural networks : the official journal of the International Neural Network Society*, 172:106135, 2024.

[71] Ziwei Peng, Kaiping Xue, Bin Zhu, Yaxuan Huang, Jian Li, Meng Hao, and Tianwei Zhang. Privtf: Efficient and privacy-preserving transformer inference with optimized protocols for secure nlp. *IEEE Transactions on Network Science and Engineering*, 2025.

[72] Chenkai Zeng, Debiao He, Qi Feng, Xiaolin Yang, and Qingcai Luo. Securegpt: A framework for multi-party privacy-preserving transformer inference in gpt. *IEEE Transactions on Information Forensics and Security*, 19:9480–9493, 2024.

[73] Patrick Yubeaton, Jianqiao Cambridge Mo, Karthik Garimella, Nandan Kumar Jha, Brandon Reagen, Chinmay Hegde, and Siddharth Garg. Truncformer: Private llm inference using only truncations. *ArXiv*, abs/2412.01042, 2024.

[74] Jinglong Luo, Yehong Zhang, Zhuo Zhang, Jiaqi Zhang, Xin Mu, Hui Wang, Yue Yu, and Zenglin Xu. Secformer: Fast and accurate privacy-preserving inference for transformer models via smpc, 2025.

[75] Andes Y. L. Kei and Sherman S. M. Chow. SHAFT: Secure, handy, accurate, and fast transformer inference. In *NDSS*, 2025.

[76] Manuel B. Santos, Dimitris Mouris, Mehmet Ugurbil, Stanislaw Jarecki, José Reis, Shubho Sengupta, and Miguel de Vega. Curl: Private LLMs through Wavelet-Encoded Look-Up Tables. In *Proceedings of the Conference on Applied Machine Learning in Information Security*, pages 1–31, Arlington, Virginia, USA, October 24–25, 2024.

[77] Ke Cheng, Yuheng Xia, Anxiao Song, Jiaxuan Fu, Wenjie Qu, Yulong Shen, and Jiaheng Zhang. Mosformer: Maliciously secure three-party inference framework for large transformers. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 4124–4138, 2025.

[78] Dake Chen, Yuke Zhang, Souvik Kundu, Chenghao Li, and Peter A. Beerel. 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)*, pages 1–9, 2023.

[79] Siqi Liu, Zhusen Liu, Donglong Chen, Wangchen Dai, Lu Zhou, Zhe Liu, Ray C. C. Cheung, and Çetin Kaya Koç. Mlformer: a high performance mpc linear inference framework for transformers. *Journal of Cryptographic Engineering*, 15, 2025.

[80] Yuke Zhang, Dake Chen, Souvik Kundu, Chenghao Li, and Peter A. Beerel. Sal-vit: Towards latency efficient private inference on vit using selective attention search with a learnable softmax approximation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5093–5102, 2023.

[81] Wenxuan Zeng, Meng Li, Wenjie Xiong, Tong Tong, Wen-Jie Lu, Jin Tan, Runsheng Wang, and Ru Huang. Mpcvit: Searching for accurate and efficient mpc-friendly vision transformer with heterogeneous attention. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5029–5040, 2023.

[82] Yanxin Liu and Qianqian Su. Pptif: Privacy-preserving transformer inference framework for language translation. *IEEE Access*, 12:48881–48897, 2024.

[83] Guang Yan, Yuhui Zhang, Zimu Guo, Lutan Zhao, Xiaojun Chen, Chen Wang, Wenhao Wang, Dan Meng, and Rui Hou. Comet: Accelerating private inference for large language model by predicting activation sparsity. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 2827–2845. IEEE, 2025.

[84] Wen jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. Bumblebee: Secure two-party inference framework for large transformers. In *NDSS*, 2025.

[85] Mehmet Ugurbil, Dimitris Mouris, Manuel B. Santos, José Cabrero-Holgueras, Miguel de Vega, and Shubho Sengupta. Fission: Distributed privacy-preserving large language model inference. Cryptology ePrint

Archive, Paper 2025/653, 2025.

[86] Wenxuan Zeng, Ye Dong, Jinjin Zhou, Jin Tan, Lei Wang, Tao Wei, Runsheng Wang, and Meng Li. Mpcache: Mpc-friendly kv cache eviction for efficient private llm inference. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[87] Zi Liang, Pinghui Wang, Ruofei Zhang, Nuo Xu, and Shuo Zhang. Merge: Fast private text generation. In *AAAI Conference on Artificial Intelligence*, 2023.

[88] Neha Jawalkar, Kanav Gupta, Arkaprava Basu, Nishanth Chandran, Divya Gupta, and Rahul Sharma. Orca: Fss-based secure training and inference with gpus. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 597–616. IEEE, 2024.

[89] Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 15718–15731. Curran Associates, Inc., 2022.

[90] Xiaoyang Hou, Jian Liu, Jingyu Li, Yuhan Li, Wen jie Lu, Cheng Hong, and Kui Ren. CipherGPT: Secure two-party GPT inference. Cryptology ePrint Archive, Paper 2023/1147, 2023.

[91] Weize Wang and Yi Kuang. Cipherformer: Efficient transformer private inference with low round complexity. *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 3054–3059, 2024.

[92] Zhicong Huang, Wen jie Lu, Cheng Hong, and Jiansheng Ding. Cheetah: Lean and fast secure two-party deep neural network inference. In *IACR Cryptology ePrint Archive*, 2022.

[93] Tianshi Xu, Lemeng Wu, Runsheng Wang, and Meng Li. Privcirnet: Efficient private inference via block circulant transformation. *ArXiv*, abs/2405.14569, 2024.

[94] Tianshi Xu, Wen-jie Lu, Jiangrui Yu, Yi Chen, Chenqi Lin, Runsheng Wang, and Meng Li. Breaking the layer barrier: Remodeling private transformer inference with hybrid {CKKS} and {MPC}. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2653–2672, 2025.

[95] Xuanqi Liu and Zhuotao Liu. Llms can understand encrypted prompt: Towards privacy-computing friendly transformers, 2023.

[96] Yuanchao Ding, Hua Guo, Yewei Guan, Weixin Liu, Jiarong Huo, Zhenyu Guan, and Xiyong Zhang. East: Efficient and accurate secure inference framework for transformer. *IEEE Transactions on Services Computing*, 18(4):2038–2046, 2025.

[97] Yancheng Zhang, Jiaqi Xue, Mengxin Zheng, Mimi Xie, Mingzhe Zhang, Lei Jiang, and Qian Lou. Cipherprune: Efficient and scalable private transformer inference. *arXiv preprint arXiv:2502.16782*, 2025.

[98] Zhengyi Li, Kang Yang, Jin Tan, Wen jie Lu, Haoqi Wu, Xiao Wang, Yu Yu, Derun Zhao, Yancheng Zheng, Minyi Guo, and Jingwen Leng. Nimbus: Secure and efficient two-party inference for transformers. *ArXiv*, abs/2411.15707, 2024.

[99] Fei Zheng, Chaochao Chen, Zhongxuan Han, and Xiaolin Zheng. Permllm: Private inference of large language models within 3 seconds under wan. *ArXiv*, abs/2405.18744, 2024.

[100] Saisai Xia, Wenhao Wang, Zihao Wang, Yuhui Zhang, Yier Jin, Dan Meng, and Rui Hou. Cryptpeft: Efficient and private neural network inference via parameter-efficient fine-tuning. In *NDSS*, 2025.

[101] Naren Dhyani, Jianqiao Cambridge Mo, Minsu Cho, Ameya Joshi, Siddharth Garg, Brandon Reagen, and Chinmay Hegde. Privit: Vision transformers for fast private inference. *ArXiv*, abs/2310.04604, 2023.

[102] Chen Song, Ruwei Huang, and Sai Hu. Private-preserving language model inference based on secure multi-party computation. *Neurocomputing*, 592:127794, 2024. Communicated by M. Gallo.

[103] Nandan Kumar Jha and Brandon Reagen. Entropy-guided attention for private llms. *arXiv preprint arXiv:2501.03489*, 2025.

[104] Deevashwer Rathee, Dacheng Li, Ion Stoica, Hao Zhang, and Raluca Popa. Mpc-minimized secure llm inference, 2024.

[105] Chenqi Lin, Tianshi Xu, Zebin Yang, Runsheng Wang, Ru Huang, and Meng Li. Fastquery: Communication-efficient embedding table query for private llms inference. *2024 61st ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2024.

[106] Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin E. Lauter, Saeed Maleki, Madan Musuvathi, and Todd Mytkowicz. Chet: an optimizing compiler for fully-homomorphic neural-network inferencing. *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019.

[107] Tim van Elsloo, Giorgio Patrini, and Hamish Ivey-Law. Sealion: a framework for neural network inference on encrypted data. *ArXiv*, abs/1904.12840, 2019.

[108] Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. ngraph-he2: A high-throughput framework for neural network inference on encrypted data. *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2019.

[109] Song Bian, Zian Zhao, Zhou Zhang, Ran Mao, Kohei Suenaga, Yier Jin, Zhenyu Guan, and Jianwei Liu. Heir: A unified representation for cross-scheme compilation of fully homomorphic computation. *IACR Cryptol. ePrint Arch.*, 2023:1445, 2023.

[110] Roshan Dathathri, Blagovesta Kostova, Olli Saarikivi, Wei Dai, Kim Laine, and Madan Musuvathi. Eva: an encrypted vector arithmetic language and compiler for efficient homomorphic computation. *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019.

[111] Huili Chen, Rosario Cammarota, Felipe Valencia, Francesco Regazzoni, and Farinaz Koushanfar. Ahec: End-to-end compiler framework for privacy-preserving machine learning acceleration. *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[112] Meghan Cowan, Deeksha Dangwal, Armin Alaghi, Caroline Trippel, Vincent T. Lee, and Brandon Reagen. Porcupine: a synthesizing compiler for vectorized homomorphic encryption. *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, 2021.

[113] Raghav Malik, Kabir Sheth, and Milind Kulkarni. Coyote: A compiler for vectorizing encrypted arithmetic circuits. *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2023.

[114] Ehud Aharoni, Allon Adir, Moran Baruch, Nir Drucker, Gilad Ezov, Ariel Farkash, Lev Greenberg, Ramy Masalha, Dov Murik, Hayim Shaul, and Omri Soceanu. Helayers: A tile tensors framework for large neural networks on encrypted data. *Proc. Priv. Enhancing Technol.*, 2023:325–342, 2020.

[115] Aleksandar Krastev, Nikola Samardzic, Simon Langowski, Srinivas Devadas, and Daniel Sanchez. A tensor compiler with automatic data packing for simple and efficient fully homomorphic encryption. *Proceedings of the ACM on Programming Languages*, 8:126 – 150, 2024.

[116] Alexander Viand, Patrick Jattke, Miro Haller, and Anwar Hithnawi. Heco: Fully homomorphic encryption compiler. In *USENIX Security Symposium*, 2022.

[117] Jiangrui Yu, Wenxuan Zeng, Tianshi Xu, Renze Chen, Yun Liang, Runsheng Wang, Ru Huang, and Meng Li. Flexhe: a flexible kernel generation framework for homomorphic encryption-based private inference. *2024 ACM/IEEE International Conference On Computer Aided Design (ICCAD)*, pages 01–09, 2024.

[118] Yongwoo Lee, Seonyeong Heo, Seonyoung Cheon, Shinnung Jeong, Changsu Kim, Eunkyung Kim, Dongyoon Lee, and Hanjun Kim. Hecate: Performance-aware scale optimization for homomorphic encryption compiler. *2022 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 193–204, 2022.

[119] Yongwoo Lee, Seonyoung Cheon, Dongkwan Kim, Dongyoon Lee, and Hanjun Kim. Elasm: Error-latency-aware scale management for fully homomorphic encryption. In *USENIX Security Symposium*, 2023.

[120] Seonyoung Cheon, Yongwoo Lee, Dongkwan Kim, Ju Min Lee, Sunchul Jung, Taekyung Kim, Dongyoon Lee, and Hanjun Kim. Dacapo: Automatic bootstrapping management for efficient fully homomorphic encryption. In *USENIX Security Symposium*, 2024.

[121] Yan Liu, Jianxin Lai, Long Li, Tianxiang Sui, Linjie Xiao, Peng Yuan, Xiaojing Zhang, Qing Zhu, Wenguang Chen, and Jingling Xue. Resbm: Region-based scale and minimal-level bootstrapping management for fhe via min-cut. *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2025.

[122] Seonyoung Cheon, Yongwoo Lee, Hoyun Youm, Dongkwan Kim, Sungwoo Yun, Kunmo Jeong, Dongyoon Lee, and Hanjun Kim. Halo: Loop-aware bootstrapping management for fully homomorphic encryption. *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2025.

[123] Long Li, Jianxin Lai, Peng Yuan, Tianxiang Sui, Yan Liu, Qing Zhu, Xiaojing Zhang, Linjie Xiao, Wenguang Chen, and Jingling Xue. Ant-ace: An fhe compiler framework for automating neural network inference. *Proceedings of the 23rd ACM/IEEE International Symposium on Code Generation and Optimization*, 2025.

[124] Austin Ebel, Karthik Garimella, and Brandon Reagen. Orion: A fully homomorphic encryption framework for deep learning. *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2023.

[125] Jiaxing He, Kang Yang, Guofeng Tang, Zhangjie Huang, Li Lin, Changzheng Wei, Ying Yan, and Wei

Wang. Rhombus: Fast homomorphic matrix-vector multiplication for secure two-party inference. *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024.

[126] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P. Wellman. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 399–414, 2018.

[127] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. Cryptology ePrint Archive, Paper 2009/616, 2009.

[128] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. Oblivious neural network predictions via MiniONN transformations. Cryptology ePrint Archive, Paper 2017/452, 2017.

[129] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. Cryptology ePrint Archive, Paper 2017/396, 2017.

[130] José Cabrero-Holgueras and Sergio Pastrana. Sok: Privacy-preserving computation techniques for deep learning. *Proceedings on Privacy Enhancing Technologies*, 2021:139 – 162, 2021.

[131] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha P. Chandrakasan. Gazelle: A low latency framework for secure neural network inference. In *IACR Cryptology ePrint Archive*, 2018.

[132] Harsh Chaudhari, Rahul Rachuri, and Ajith Suresh. Trident: Efficient 4PC framework for privacy preserving machine learning. Cryptology ePrint Archive, Paper 2019/1315, 2019.

[133] M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin Lauter, and Farinaz Koushanfar. XONN: XNOR-based oblivious deep neural network inference. Cryptology ePrint Archive, Paper 2019/171, 2019.

[134] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A cryptographic inference service for neural networks. Cryptology ePrint Archive, Paper 2020/050, 2020.

[135] Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 2019:26 – 49, 2019.

[136] Qiao Zhang, Chunsheng Xin, and Hongyi Wu. Gala: Greedy computation for linear algebra in privacy-preserved neural networks. *ArXiv*, abs/2105.01827, 2021.

[137] Arpita Patra, T. Schneider, Ajith Suresh, and Hossein Yalame. Aby2.0: Improved mixed-protocol secure two-party computation. *IACR Cryptol. ePrint Arch.*, 2020:1225, 2020.

[138] Anders Dalskov, Daniel E. Escudero, and Marcel Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. *IACR Cryptol. ePrint Arch.*, 2020:1330, 2020.

[139] Payman Mohassel and Peter Rindal. Aby3: A mixed protocol framework for machine learning. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[140] Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. Swift: Super-fast and robust privacy-preserving machine learning. *IACR Cryptol. ePrint Arch.*, 2020:592, 2020.

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

[142] Zahra Ghodsi, Akshaj Kumar Veldanda, Brandon Reagen, and Siddharth Garg. Cryptonas: Private inference on a relu budget. *ArXiv*, abs/2006.08733, 2020.

[143] Fabian Boemer, Yixing Lao, and Casimir Wierzynski. ngraph-he: a graph compiler for deep learning on homomorphically encrypted data. *Proceedings of the 16th ACM International Conference on Computing Frontiers*, 2018.

[144] Tim van Elsloo, Giorgio Patrini, and Hamish Ivey-Law. Sealion: a framework for neural network inference on encrypted data. *ArXiv*, abs/1904.12840, 2019.

[145] Najwa Aaraj, Abdelrahaman Aly, Tim Güneysu, Chiara Marcolla, Johannes Mono, Rogerio Paludo, Iván Santos-González, Mireia Scholz, Eduardo Soria-Vazquez, Victor Sucasas, and Ajith Suresh. Fanng-mpc: Framework for artificial neural networks and generic mpc. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025:1–36, 2024.

[146] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J. Kusner, and Adrià Gascón. Quotient: Two-party secure neural network training and prediction. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[147] Nishanth Chandran, Divya Gupta, Sai Lakshmi Bhavana Obbattu, and Akash Shah. Simc: Ml inference secure against malicious clients at semi-honest cost. In *IACR Cryptology ePrint Archive*, 2021.

[148] Ye Dong, Xiaojun Chen, Weizhan Jing, Kaiyun Li, and

Weiping Wang. Meteor: Improved secure 3-party neural network inference with reducing online communication costs. *Proceedings of the ACM Web Conference 2023*, 2023.

[149] Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, and Tianwei Zhang. Fastsecnet: An efficient cryptographic framework for private neural network inference. *IEEE Transactions on Information Forensics and Security*, 18:2569–2582, 2023.

[150] Christopher Harth-Kitzerow, Yongqin Wang, Rachit Rajat, Georg Carle, and Murali Annavaram. Pigeon: A framework for private inference of neural networks. *IACR Cryptol. ePrint Arch.*, 2024:1371, 2024.

[151] Fengrun Liu, Xiang Xie, and Yu Yu. Scalable multiparty computation protocols for machine learning in the honest-majority setting. In *USENIX Security Symposium*, 2024.

[152] Robert Podschwadt, Daniel Takabi, Peizhao Hu, Mohammad Hossein Rafiei, and Zhipeng Cai. A survey of deep learning architectures for privacy-preserving machine learning with fully homomorphic encryption. *IEEE Access*, 10:117477–117500, 2022.

[153] Luis Bernardo Pulido-Gaytan, Andrei Nikolaevitch Tchernykh, Jorge M. Cortés-Mendoza, Mikhail Grigoryevich Babenko, and Gleb I. Radchenko. A survey on privacy-preserving machine learning with fully homomorphic encryption. In *Latin American Conference on High Performance Computing*, 2020.

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

[155] Kapil Tiwari, Samiksha Shukla, and Jossy P. George. A systematic review of challenges and techniques of privacy-preserving machine learning. 2021.

[156] Guowen Xu, Xingshuo Han, Tianwei Zhang, Shengmin Xu, Jianting Ning, Xinyi Huang, Hongwei Li, and Robert H. Deng. Simc 2.0: Improved secure ml inference against malicious clients. *IEEE Transactions on Dependable and Secure Computing*, 21(4):1708–1723, 2024.

[157] Yansong Zhang, Xiaojun Chen, Ye Dong, Qinghui Zhang, Rui Hou, Qiang Liu, and Xudong Chen. Mdsonic: Maliciously-secure outsourcing neural network inference with reduced online communication. *IEEE Transactions on Information Forensics and Security*, 20:3534–3549, 2025.

[158] Yu Zheng, Qizhi Zhang, Sherman S. M. Chow, Yuxiang Peng, Sijun Tan, Lichun Li, and Shan Yin. Secure softmax/sigmoid for machine-learning computation. *Proceedings of the 39th Annual Computer Security Applications Conference*, 2023.

[159] Lijing Zhou, Bingsheng Zhang, Ziyu Wang, Tianpei Lu, Qingrui Song, Su Zhang, Hongrui Cui, and Yu Yu. On probabilistic truncation in privacy-preserving machine learning. In *AAAI Conference on Artificial Intelligence*, 2025.

[160] Marcel Keller. Mp-spdz: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 1575–1590, 2020.

[161] Jean-Luc Watson, Sameer Wagh, and Raluca A. Popa. Piranha: A gpu platform for secure computation. In *IACR Cryptology ePrint Archive*, 2022.

[162] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, T. Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018.

[163] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 420–432, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[164] Daniel Demmler, T. Schneider, and Michael Zohner. Aby - a framework for efficient mixed-protocol secure two-party computation. In *Network and Distributed System Security Symposium*, 2015.

[165] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. CrypTFlow2: Practical 2-party secure inference. Cryptology ePrint Archive, Paper 2020/1002, 2020.

[166] Brian Knott, Shobha Venkataraman, Awni Y. Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. *ArXiv*, abs/2109.00984, 2021.

[167] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow: Secure tensorflow inference. *2020 IEEE Symposium on Security and Privacy (SP)*, pages 336–353, 2019.

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

[169] Junming Ma, Yancheng Zheng, Jun Feng, Derun Zhao,

Haoqi Wu, Wenjing Fang, Jin Tan, Chaofan Yu, Benyu Zhang, and Lei Wang. {SecretFlow-SPU}: A performant and {User-Friendly} framework for {Privacy-Preserving} machine learning. In *2023 USeNIX annual technical conference (USeNIX ATC 23)*, pages 17–33, 2023.

[170] Zhenyu Guan, Ran Mao, Qianyun Zhang, Zhou Zhang, Zian Zhao, and Song Bian. Autohog: Automating homomorphic gate design for large-scale logic circuit evaluation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(7):1971–1983, 2024.

[171] Wenjie Qu, Yijun Sun, Xuanming Liu, Tao Lu, Yanpei Guo, Kai Chen, and Jiaheng Zhang. zkgpt: An efficient non-interactive zero-knowledge proof framework for llm inference. In *34st USENIX Security Symposium (USENIX Security 25)*, 2025.

[172] Shai Halevi and Victor Shoup. Design and implementation of HElib: a homomorphic encryption library. Cryptology ePrint Archive, Paper 2020/1481, 2020.

[173] Alberto Ibarrondo and Alexander Viand. Pyfhel: Python for homomorphic encryption libraries. In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 11–16, 2021.

[174] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library-seal v2. 1. In *International conference on financial cryptography and data security*, pages 3–18. Springer, 2017.

[175] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. Openfhe: Open-source fully homomorphic encryption library. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, WAHC'22, pages 53–63, New York, NY, USA, 2022. Association for Computing Machinery.

[176] Zama. TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data, 2022. https://github.com/zama-ai/tfhe-rs.

[177] Lattigo v6. Online: https://github.com/tuneinsight/lattigo, August 2024. EPFL-LDS, Tune Insight SA.

[178] Ali Şah Özcan and Erkay Savaş. HEonGPU: a GPU-based fully homomorphic encryption library 1.0. Cryptology ePrint Archive, Paper 2024/1543, 2024.

[179] Xiao Wang, Alex J Malozemoff, and Jonathan Katz. Emp-toolkit: Efficient multiparty computation toolkit, 2016.

[180] Marcel Keller. Mp-spdz: A versatile framework for multi-party computation. *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

Appendix

A BERT architecture

In most cases, BERT uses the encoder-only mode, exemplified by models such as BERT-Base, as shown in Fig. 6. The following section discusses the key operators of a single BERT-Base block.

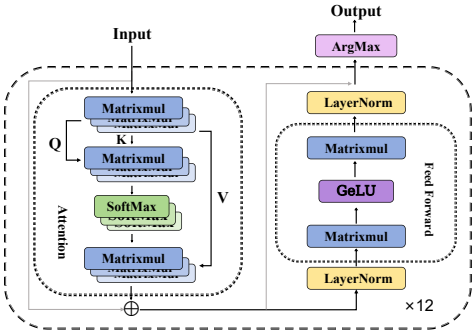


Figure 6: The architecture of BERT-Base.

Table 4: Statistics on extensibility of PPML frameworks to large model private inference.

Extensibility	PPML Works
×	[41], [125], [126], [127], [128], [129], [10], [130] [131], [132], [133], [134], [135], [136], [137], [138] [139], [140], [141], [142], [143], [106], [144], [108] [145], [146], [147], [11], [148], [149], [150], [92] [151], [152], [153], [154], [155], [156], [9], [157] [158], [159], [160], [161], [162], [163]
✓	[164], [165], [166],[167], [168], [169]

The left column indicates whether the technique can be extended (✓) or not (×) to transformer-based large models; the right column lists representative PPML works.

Attention Layer. Given matrices **Q** (query), **K** (key), **V** (value) in the latent representation, the attention layer can be formalized as Eq. (1) :

$$Attention(\mathbf{Q},\mathbf{K},\mathbf{V}) = SoftMax\left(\frac{\mathbf{Q}\otimes\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \quad (1)$$

Table 5: Open-source implementations and libraries.

Library	Year	Support	Language	w. GPU	Repository Link
HElib [172]	2020	BGV, CKKS	C++	YES	https://github.com/homenc/HElib
Pyfhel [173]	2021	BGV, BFV, CKKS	Python & C++	NO	https://github.com/ibarrond/Pyfhel
SEAL [174]	2018	BGV, BFV, CKKS	C++	YES$^\sharp$	https://github.com/microsoft/SEAL
OpenFHE [175]	2022	BGV, BFV, CKKS	C++, Python	NO	https://github.com/openfheorg
TFHE-rs [176]	2022	TFHE	C, Rust	NO	https://github.com/zama-ai/tfhe-rs
Lattigo [177]	2024	BGV, BFV, CKKS	Go	NO	https://github.com/tuneinsight/lattigo
HeOnGPU [178]	2024	BGV, BFV, CKKS, TFHE	C++	YES	https://github.com/Alisah-Ozcan/HEonGPU
ABY [164]	2015	MPC	C++	NO	https://github.com/encryptogroup/ABY
EMP-toolkit [179]	2016	MPC	C++	NO	https://github.com/emp-toolkit
ABY^3 [139]	2018	MPC	C++	NO	https://github.com/ladnir/aby3
CrypTen [166]	2019	MPC	Python	YES	https://github.com/facebookresearch/CrypTen
MP-SPDZ [180]	2016	MPC, HE	Python & C++	NO	https://github.com/data61/MP-SPDZ
EzPC [168]	2019	MPC, HE	C++	YES	https://github.com/mpc-msri/EzPC
Secretflow [169]	2023	MPC, HE	Python & C++	SOON	https://github.com/secretflow/secretflow

Note: w. GPU indicates whether a library supports GPU acceleration: "YES" supported, "SOON" under development or coming soon, "NO" indicates no GPU support. "$\sharp$" indicates that open-source GPU support exists, but is not natively supported by the library.

where $\mathbf{Q},\mathbf{K},\mathbf{V} \in \mathbb{Q}^{d_s \times d_k}$. The SoftMax function applies to each row vector of a matrix. For a vector $\mathbf{x} \in \mathbb{Q}^{d_s}$. The SoftMax on $\mathbf{x}$ returns a vector $\mathbf{y} \in \mathbb{Q}^{d_s}$ can be described as $\mathbf{y} = SoftMax(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k-1} e^{x_j}}$.

The multi-head attention can be further extended by employing the attention mechanism. Considering h different attention mechanisms with weight matrices $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ for $i = 1, ..., h$, we have

$$\mathbf{h}_i = Attention(\mathbf{Q} \otimes \mathbf{W}_i^Q, \mathbf{K} \otimes \mathbf{W}_i^K, \mathbf{V} \otimes \mathbf{W}_i^V), \quad (2)$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{Q}^{d_k \times d_m}$, and $h_i \in \mathbb{Q}^{d_s \times d_m}$. Then, the multi-head attention can be formalized as Eq. (3):

$$MH(\mathbf{Q},\mathbf{K},\mathbf{V}) = (\mathbf{h}_1 || \mathbf{h}_2 || \ldots || \mathbf{h}_h) \odot \mathbf{W}^O, \quad (3)$$

where "||" denotes the horizontal concatenation of matrices, $\mathbf{W}^O \in \mathbb{Q}^{hd_k \times d_m}$, and $\mathrm{MH}(\mathbf{Q},\mathbf{K},\mathbf{V}) \in \mathbb{Q}^{d_s \times d_m}$

LayerNorm Layer. Let $\mathbf{x} := (x_1, x_2, ..., x_k)$ be a vector representation of an input of size d_k to normalization layers. LayerNorm re-centers and re-scales input $\mathbf{x} \in \mathbb{Q}^{\mathbf{d_k}}$ as Eq.(4):

$$LayerNorm(x_i) = \gamma \cdot \frac{x_i - \mu}{\sigma} + \beta, \quad (4)$$

where $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$, and $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$. μ and σ are the mean and standard deviation of $\mathbf{x}$.

Feed-forward Layer (FFN). The FFN consists of two fully connected layers and is separated by a GeLU activation. Let $\mathbf{X} \in \mathbb{Q}^{d_{s \times d_m}}$ be an input, where d_s is the sequence length. The formula is as Eq. (5):

$$FFN(\mathbf{X}) = GeLU(\mathbf{X} \otimes \mathbf{W}^{F1} \oplus \mathbf{B}^{F1}) \otimes \mathbf{W}^{F2} \oplus \mathbf{B}^{F2}, \quad (5)$$

where $\mathbf{W}^{F1} \in \mathbb{Q}^{d_m \times d_f}$, $\mathbf{W}^{F2} \in \mathbb{Q}^{d_m} \times d_f$ are the weight matrices and $\mathbf{B}^{F1} \in \mathbb{Q}^{d_f}$ $\mathbf{B}^{F2} \in \mathbb{Q}^{d_m}$ are the bias vectors for the two layers of FFN. GeLU is the Gaussian Error Linear Unit activation function, which returns a vector $\mathbf{y} \in \mathbb{Q}^{d_k}$ for the input vector $\mathbf{x} \in \mathbb{Q}^{d_k}$, as the Eq. (6):

$$GeLU(x_i) = x_i \cdot \Phi(x_i) = x_i \cdot \frac{1}{2}\left[1 + erf\left(\frac{x_i}{\sqrt{2}}\right)\right], \quad (6)$$

, where $\Phi(x_i)$ is the standard Gaussian cumulative distribution function.

ArgMax Layer. The ArgMax operation is typically used in the final classification stage to determine the predicted class. Given an input vector $\mathbf{x} \in \mathbb{Q}^{d_k}$, the ArgMax function returns the index of the maximum element, as shown in Eq. (7):

$$ArgMax(\mathbf{x}) = \arg \max_{i \in \{0,1,\ldots,d_k-1\}} x_i, \quad (7)$$

where x_i denotes the i-th element of the vector $\mathbf{x}$. The output is a scalar index $j \in \{0, 1, \ldots, d_k - 1\}$ such that $x_j \geq x_i$ for all $i \neq j$. For a matrix $\mathbf{X} \in \mathbb{Q}^{d_s \times d_k}$, the ArgMax operation can be applied row-wise, returning an index vector $\mathbf{y} \in \mathbb{Z}^{d_s}$, and $y_i = \arg\max_{j \in \{0,1,\ldots,d_k-1\}} X_{i,j}$, where $X_{i,j}$ represents the element in the i-th row and j-th column of $\mathbf{X}$.

B Evaluated Setting

Model Architectures for Figure 3, Figure 4 and Figure 5. For the experiments, we measure the performance on BERT-Base [26] (the model architecture see A) and denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A. The size of feed-forward is $4H$, i.e., 3072 for the $H = 768$ and $L = 12$,

Test Environment for Figure 3, Figure 4 and Figure 5. The experiments were conducted on a rack-mounted server equipped with an Intel(R) Xeon(R) Gold 6238R CPU operating at 2.20 GHz and 377 GB of RAM, and accelerated by an NVIDIA A10 GPU. Network bandwidth and latency were regulated using the Linux Traffic Control ('tc') utility. We evaluated the system under two network configurations: a local area network (LAN) environment with 3.7 Gbps bandwidth and 0.04 ms round-trip latency, and a wide area network (WAN) environment with 100 Mbps bandwidth and 80 ms round-trip latency.

Threat Model & Parties Numebr for Figure 3, Figure 4 and Figure 5. MosFormer adopts a malicious model, other schemes assume semi-honest models. SIGMA, SHAFT, and

Table 6: Summary Overview and Trends of Pure-MPC Frameworks

Category	Core Research Content	Key Technical Features	Representative Literature
Core Framework	Building end-to-end, efficient, and accurate MPC inference systems	High-precision approximation, protocol optimization, compatibility, security model partition	PUMA [14], SecFormer [74], Mosformer [77], Privformer [13], SHAFT [75], SecureGPT [72], Curl [76]
Component Optimization	Deep optimization for specific bottlenecks in inference chains (sparsity, caching, redundant computation)	Activation sparsity prediction, cache management, computation graph optimization	Comet [83], MPCache [86], MERGE [87]
Architecture Modification	Designing new neural network architectures friendly to MPC (especially ViT)	Neural architecture search, attention mechanism innovation, function approximation	MPC-ViT [81], RNA-ViT [78], SAL-ViT [80], Cryptpeft [100]
Methodological Innovation	Proposing new low-level computational architectures and efficiency sharing tools	function secret sharing (SIGMA [50])	Fission [85], SIGMA [50, 88]

Table 7: Comparison of HE-based non-interactive private inference protocols (time in seconds).

Frameworks	Models	Platform	Tokens	Time	Impl.
THE-X [12]	BERT-Tiny	CPU (4)	128	4700	P
PolyTransformer [54]	BERT-like	CPU (64) + GPU	128	211	P
Power-Softmax [55]	RoBERTa-like	CPU + GPU	128	400	P
EncryptedLLM [57]	-	CPU (8)	40	600	P
Tricycle [58]	BERT-Tiny	CPU (128)	44	322	P
THOR [59]	BERT-Base	GPU	128	602.26	P
Zhang et al. [63]	ChatGLM2	CPU	128	1.61	P
NEXUS [17]‡	BERT-Base	CPU (32)	128	4822.13	R
Leporati et al. [56]	BERT-Tiny	CPU (8)	40	227	R
Powerformer [60]	BERT-Base	GPU	128	900	R
MOAI [61]	BERT-Base	GPU	128	>574	R
	BERT-Tiny	GPU	128	>141	R
Arion [62]‡	BERT-Base	CPU	128	>7200	R
	BERT-Tiny	CPU	40	622	R
Euston [64]	BERT-Base	GPU	128	979	R
	LLaMA-3-8B	GPU	128	3846	R

Note: CPU (n) denotes the number of threads used. **Impl** column: "P" indicates the result is taken directly from the original paper, and 'R' means the result is reproduced in this work. The running time marked with "‡" denotes that the data is derived by summing the time of individual protocols. "$>$" indicates a crash during program execution.

SecFormer are 2-PC+dealer setting; MosFormer, PUMA, and Ditto are 3-PC setting; BOLT, IRON, and BLB are 2-PC setting. Powerformer, Nexus, and Euston adopt C/S settings.

Experimental configuration for Tables 9-12. First, all experiments reported in Tables 9-12 are conducted on a unified hardware platform equipped with an Intel(R) Xeon(R) Gold 6238R CPU at 2.20 GHz, 377 GB RAM, and an NVIDIA A10 GPU, with network bandwidth and latency controlled via the Linux Traffic Control (tc) utility. Since fully reproducing all baseline systems is challenging, we follow the original papers' model settings and network configurations when re-implementing these methods.

Code Reproducibility Review. We quantified three key implementation characteristics: the proportion of projects that provide open-source implementations, the proportion that support GPU acceleration, and, among open-source projects, the fraction that are executable. These values were 50.8% (31/61), 32.8% (20/61), and 47.5% (29/61), respectively.

Table 8: Literature Related to the private transformer-based model inference (Part I)

Methods	Venue	Year	Scenario	Threat model	Crypto Tools				Backend
					GC	OT	SS	HE	
Cryptpeft [100]	NDSS	2026	2-PC	Semi-honest	△	○	◧	◇	CrypTen
Euston [64]	ePrint	2026	C/S	Semi-honest	△	○	□	◆	phantom-fhe/SEAL
Fission [85]	ePrint	2025	2-PC	Semi-honest	△	○	◧	◇	CrypTen
Arion [62]	ePrint	2025	C/S	Semi-honest	△	○	□	◆	Lattigo
Garimella et al. [65]	arXiv	2025	C/S	Semi-honest	△	○	□	◆	Orion
Zhang et al. [63]	arXiv	2025	C/S	Semi-honest	△	○	□	◆	SEAL
MPCache [86]	NeurIPS	2025	2/3-PC	Semi-honest	△	●	◧	◇	SecretFlow
Comet [83]	IEEE S&P	2025	3-PC	Semi-honest	△	●	◧	◇	CrypTen
Lu et al. [36]	NDSS	2025	2-PC	Semi-honest	△	●	◧	◇	custom
SecFormer [74]	arXiv	2025	2-PC-Dealer	Semi-honest	△	●	◧	◇	CrypTen
Mosformer [77]	SIGSAC	2025	3-PC	Malicious	△	○	■	◇	custom
THOR [59]	SIGSAC	2025	C/S	Semi-honest	△	○	□	◆	Liberate.FHE
MOAI [61]	ePrint	2025	C/S	Semi-honest	△	○	□	◆	SEAL
Tricycle [58]	ePrint	2025	C/S	Semi-honest	△	○	□	◆	SEAL
CipherPrune [97]	ICLR	2025	2-PC	Semi-honest	△	●	◧	◆	EzPC, SEAL
BumbleBee [84]	NDSS	2025	C/S	Semi-honest	△	●	◧	◆	SecretFlow
EAST [96]	Trans. Serv. Comput	2025	2-PC	Semi-honest	△	○	◧	◆	EMP, EzPC, SEAL
BLB [94]	USENIX	2025	2-PC	Semi-honest	△	●	◧	◆	EzPC, SEAL
Powerformer [60]	ACL	2025	C/S	Semi-honest	△	○	□	◆	Lattigo
NEXUS [17]	NDSS	2025	C/S	Semi-honest	△	○	□	◆	SEAL
EncryptedLLM [57]	ICML	2025	C/S	Semi-honest	△	○	□	◆	OpenFHE
SHAFT [75]	NDSS	2025	2-PC-Dealer	Semi-honest	△	○	◧	◇	CrypTen
MLFormer [79]	J Crypt Eng	2025	2-PC-Dealer	Semi-honest	△	●	◧	◇	CrypTen
Aero [103]	arXiv	2025	2-PC	Semi-honest	△	●	■	◆	BumbleBee
PUMA [14]	Security and Safety	2025	3-PC	Semi-honest	△	○	◨	◇	SecretFlow
FASTLMPI [53]	IEEE TIFS	2025	C/S	Semi-honest	△	○	◧	◆	EzPC
PrivTF [71]	Trans. Netw. Sci. Eng.	2025	C/S	Semi-honest	▲	●	■	◇	SCI
Donghwan et al. [66]	ICLR	2025	C/S	Semi-honest	△	○	□	◆	HEaaN
ENSI [67]	arXiv	2025	C/S	Semi-honest	△	○	□	◆	OpenFHE
MERGE [87]	AAAI	2024	2-PC	Semi-honest	△	○	◧	◇	CrypTen
MARILL [104]	arXiv	2024	2-PC-Dealer	Semi-honest	△	○	◧	◇	/
Curl [76]	ePrint	2024	2-PC-Dealer	Semi-honest	△	○	◧	◇	CrypTen
Ditto [16]	ICML	2024	3-PC	Semi-honest	△	○	◨	◇	SecretFlow
Song et al. [102]	Neurocomputing	2024	3-PC	Semi-honest	△	○	◧	◇	CrypTen
PermLLM [99]	arXiv	2024	2-PC-Dealer	Semi-honest	△	○	◧	◆	Pyfhel
Truncformer[73]	arXiv	2024	2-PC	Semi-honest	▲	●	◧	◇	CrypTen
CipherFormer [91]	CSCWD	2024	2-PC	Semi-honest	▲	○	□	◆	Gazelle-BFV
FastQuery [105]	DAC	2024	C/S	Semi-honest	△	○	◧	◆	Cheetah/Iron
PolyTransformer [54]	ICML	2024	C/S	Semi-honest	△	○	□	◆	/
Leporati et al. [56]	IWSPA	2024	C/S	Semi-honest	△	○	□	◆	OpenFHE
PrivCirNet [93]	NeurIPS	2024	C/S	Semi-honest	▲	●	□	◆	/
SecBERT [70]	Neural Networks	2024	2-PC-Dealer	Semi-honest	△	○	◧	◇	ABY
Nimbus [98]	NeurIPS	2024	2-PC	Semi-honest	△	●	◧	◆	SecretFlow
BOLT [33]	IEEE S&P	2024	C/S	Semi-honest	△	○	◧	◆	EzPC,SEAL
SecureGPT [72]	IEEE TIFS	2024	n-PC	Semi-honest	△	●	◧	◇	emp-toolkit
SIGMA [50]	PETS	2024	2-PC-Dealer	Semi-honest	△	○	■	◇	EzPC
PPTIF [82]	IEEE Access	2024	3-PC	Malicious	△	○	◨	◇	MP-SPDZ
Power-Softmax [55]	arXiv	2024	C/S	Semi-honest	△	○	□	◆	HEaaN
SecureTLM [69]	Information Sciences	2024	2-PC-Dealer	Semi-honest	△	○	◧	◆	EzPC
PriViT [101]	arXiv	2023	2-PC	Semi-honest	△	●	◧	◇	SecretFlow
Liu et al. [95]	arXiv	2023	2-PC	Semi-honest	△	●	◧	◆	OpenCheetah
MPCFormer [15]	ICLR	2023	2-PC-Dealer	Semi-honest	△	●	□	◇	CrypTen
SAL-ViT [80]	ICCV	2023	2-PC	Semi-honest	△	●	◧	◇	CrypTen
MPC-ViT [81]	ICCV	2023	2-PC	Semi-honest	△	●	◧	◇	SecretFlow
RNA-ViT [78]	ICCAD	2023	2-PC	Semi-honest	△	●	◧	◇	CrypTen
CipherGPT [90]	ePrint	2023	2-PC	Semi-honest	△	○	◧	◆	SEAL
Privformer [13]	EuroS&P	2023	3-PC	Semi-honest	△	○	◨	◇	Falcon
Primer [52]	DAC	2023	2-PC	Semi-honest	▲	○	□	◆	/
Iron [89]	NeurIPS	2022	2-PC	Semi-honest	△	○	◧	◆	SEAL,EzPC
Wang et al. [18]	ISPASS	2022	2-PC-Dealer	Semi-honest	△	○	◧	◇	CrypTen
THE-X [12]	ACL	2022	2-PC	Semi-honest	△	○	□	◆	SEAL

Note: The Crypto Tools subcategories use four symbols to represent their usage status: **GC** (Garbled Circuits, △ for not used and ▲ for used), **OT** (Oblivious Transfer, ○ for not used and ● for used), **SS** (Arithmetic Secret Sharing and Replicated Secret Sharing, □ for not used and ■ for Function Secret Sharing and ◧ for Arithmetic Secret Sharing and ◨ for Replicated Secret Sharing), and **HE** (Homomorphic Encryption, ◇ for not used and ◆ for used).

Table 9: Resource requirements, tasks performed, dataset performance, and running time (part I).

Method	Model	Dataset	Comm.	Comm. Set.	Performance			Accessibility			Impl.	Time
					Plain	Enc.	Loss ↓	Code	Exe.	w/.GPU		
Cryptpeft [100]	ViT-Base-16	CIFAR-10	0.56 GB	(1 Gbps, 0.5 ms)	-	97.29%	-	✓	✓	✓	R	11.78 s
		CIFAR-100			-	85.63%	-					
		Food-101			-	84.84%	-					
		SVHN			-	91.82%	-					
		Flowers-102			-	92.60%	-					
Euston [64]	BERT-Base	RTE	37.24 MB	(10 Gbps, -)	70.04%	70.04%	0%	✓	✓	✓	R	3846 s
		SST-2			93.23%	93.12%	0.11%					
		QNLI			91.25%	91.10%	0.15%					
	LLaMA-3-8B	RTE	-		82.75%	81.24%	1.51%					979.32 s
		SST-2			94.94%	94.46%	0.48%					
		QNLI			90.70%	90.38%	0.32%					
Fission [85]	BERT-Base	QNLI	1.268GB	(-, - ms)	90.9%	90.86%	0.04%	✓	✓	✓	R	2.21 s
		SST-2			92.3%	91.7%	0.62%					
	GPT-2	Lambada	1.530 GB	(-, - ms)	35.5%	35.4%	0.1%					2.78 s
Arion [62]	BERT-Tiny	QNLI	-	(-, - ms)	96.06%	95.84%	0.18%	✓	✓	✗	R	622 s
		RTE			-	-	0.01%					
		SST-2				-	0.8%					
Zhang et al.[63]	ChatGLM2-6B	-	-	(-, - ms)	-	-	-	✗	✗	✗	P	1.61 s
Garimella et al. [65]	-	-	-	(-, - ms)	-	-	-	✗	✗	✗	P	-
MPCache [86]	-	-	-	(-, - ms)	-	-	-	✗	✗	✗	P	-
Comet[83]	LLaMA-2-7B	PIQA	659.00 GB	(5 Gbps, -)	78.40%	76.80%	1.60%	✗	✗	✗	P	1464.00 s
		BoolQ			62.50%	60.40%	2.10%					
		MMLU			44.50%	42.80%	1.70%					
		COPA			81.30%	78.50%	2.80%					
		Lambada			66.20%	63.40%	2.80%					
Lu et al. [36]	GPT-2	CIFAR-10	-	(9.4, 0.05- ms)	67.41%	66.67%	0.74%	✗	✗	✓	P	42.71 s
SecFormer[74]	BERT-Base	RTE	23.59 GB	(10 Gbps, -)	69.70%	69.00%	0.70%	✓	✓	✓	R	379.87 s
		MRPC			90.30%	89.20%	1.10%					
		CoLA			57.80%	57.10%	0.70%					
		STS-B			89.10%	87.40%	1.70%					
		QNLI			91.70%	91.20%	0.50%					
	BERT-Large	RTE	50.36 GB		75.50%	72.60%	2.90%					147.2 s
		MRPC			90.30%	88.70%	1.60%					
		CoLA			61.70%	61.30%	0.40%					
		STS-B			90.20%	89.20%	1.00%					
		QNLI			92.40%	92.00%	0.40%					
MosFormer[77]	BERT-Base	QNLI	4.60 GB	(1 Gbps, 0.5 ms)	90.60%	89.10%	1.70%	✓	✓	✗	R	228.41 s
		RTE			69.70%	68.60%	1.60%					
		STS-B			86.10%	87.00%	-1.00%					
THOR[59]	GPT2-Base	WikiText-103	3.07 GB	-	37.50%	38.40%	2.40%	✓	✗	✓	P	160.10 s
MOAI[61]	BERT-Base	MRPC	-	-	85.29%	84.80%	0.49%	✓	✗	✓	P	602.26 s
		SST-2	-	>574 s	-	93.90%	-	✓	✗	✓	-	-
		QNLI			-	90.70%	-					
		RTE			-	65.70%	-					
Tricycle[58]	BERT-Tiny	-	-	-	-	-	-				P	> 141 s
	BERT-Tiny	SST	-	-	-	-	-	✗	✗	✗	P	322.0 s
CipherPrune[97]	BERT-Base	MNLI	47.45 GB	(3 Gbps, 0.8 ms)	-	84.74%	-	✓	✗	✗	P	79.1 s
		QNLI			-	90.17%	-					
		STS-2			-	92.75%	-					
		MRPC			-	90.45%	-					
BumbleBee[84]	BERT-Base	RTE	6.40 GB	(1 Gbps, 0.5 ms)	70.04%	70.04%	0.00%	✓	✓	✓	R	133.45 s
		QNLI			90.30%	90.20%	0.10%					
		CoLA			61.57%	60.82%	0.75%					
	ViT-Base	ImageNet	2.77 GB		73.17%	72.57%	0.60%					1142.81 s
EAST[96]	BERT-Base	SST-2	163.6 GB	(377 Mbps, 0.8 ms)	-	-	-	✗	✗	✗	P	6526.4 s ‡
BLB[94]	BERT-Base	MRPC	3.00 GB	(1 Gbps, 0.3 ms)	88.80%	88.50%	0.30%	✓	✓	✓	R	452.60 s
		RTE			69.30%	69.70%	-0.50%					
		SST-2			93.20%	93.00%	0.20%					
		QNLI			91.30%	91.20%	0.10%					
	BERT-Large	MRPC	7.80 GB		89.20%	89.10%	0.10%					1120.37 s
		RTE			70.40%	70.00%	0.40%					
		SST-2			92.50%	92.40%	0.10%					
		QNLI			91.90%	91.70%	0.20%					
	GPT2-Base	MRPC	1.50 GB		82.70%	82.50%	0.20%					407.26 s
		RTE			66.10%	66.40%	-0.30%					
		SST-2			89.70%	89.90%	-0.20%					
		QNLI			88.90%	88.90%	0.00%					
Powerformer [60]	BERT-Base	RTE	-	-	69.43%	69.92%	-0.49%	✓	✓	✓	R	900 s
		MRPC			87.01%	86.76%	0.52%					
		SST-2			92.13%	92.09%	0.04%					

Note: "✓" and "✗" indicate whether the code and executable are open-source and available, respectively. In the **Impl.** column, "P" indicates the data is taken directly from the original paper, and "R" means the result is reproduced in this work. "‡" denotes that the data is derived by summing the running time of individual protocols.

Table 10: Resource requirements, tasks performed, dataset performance, and runtime (part II).

Method	Model	Dataset	Comm.	Comm. Set.	Performance			Accessibility			Impl.	Time
					Plain	Enc.	Loss ↓	Code	Exe.	w/.GPU		
NEXUS [17]	BERT-Base	RTE	0.16 GB	(3 Gbps, 0.8 ms)	70.04%	69.88%	0.16%	✓	✓	✓	R	4822.13 s
		SST-2			92.36%	92.11%	0.25%					
		QNLI			90.30%	89.90%	0.40%					
	LLaMA-3-8B	RTE	-		82.75%	81.24%	1.51%					> 12859.01 s
		SST-2			94.94%	94.46%	0.48%					
		QNLI			90.70%	90.20%	0.50%					
EncryptedLLM[57]	GPT2-Base	ARC	-	-	43.80%	42.90%	0.90%	✗	✗	✓	P	-
		HEllaSwag			28.90%	28.10%	0.80%					
		Social IQA			36.60%	37.40%	-0.80%					
		ANLI			34.90%	34.80%	0.10%					
		Wic			49.20%	51.10%	-1.90%					
	GPT2-Medium	ARC	-	-	49.00%	48.20%	0.80%					600 s
		HEllaSwag			33.30%	31.90%	1.40%					
		Social IQA			39.10%	39.00%	0.10%					
		ANLI			34.90%	34.10%	0.80%					
		Wic			50.00%	49.10%	0.90%					
	GPT2-Large	ARC	-	-	53.10%	50.70%	2.40%					-
		HEllaSwag			36.30%	32.10%	4.20%					
		Social IQA			39.50%	39.10%	0.40%					
		ANLI			33.30%	33.30%	0.00%					
		Wic			49.60%	48.10%	1.50%					
SHAFT[75]	BERT-Base	QNLI	10.46 GB	(3 Gbps, 4 ms)	90.77%	90.38%	0.39%	✓	✓	✓	R	73.61 s
		SST-2			92.66%	66.40%	0.46%					
		CoLA			57.78%	56.85%	0.93%					
MLFormer [79]	ViT	Fashion_Mnist	-	-	83.66%	83.67%	-0.01%	✗	✗	✓	P	-
Aero [103]	GPT-2	CodeParrot	-	(100 Mbps, 80 ms)	-	-	-	✓	✗	✓	P	285.6 s
		Languini			-	-	-					1166.4 s
PUMA [14]	BERT-Base	CoLA	32.71 GB	(3.76 Gbps, 0.8 ms)	61.6%	61.3%	0.3%	✓	✓	✗	R	282.73 s
		RTE			70.0%	70.0%	0%					
		QNLI			91.6%	91.6%	0%					
	RoBERTa-Base	CoLA	-		62.9%	61.8%	1.1%					-
		RTE			80.5%	80.5%	0%					
		QNLI			92.0%	91.8%	0.2%					
	BERT-Large	CoLA	86.9 GB		68.6%	69.0 %	-0.4%					690.47 s
		RTE			75.5%	74.7%	0.8%					
		QNLI			92.2%	91.8%	0.4%					
FASTLMPI [53]	BERT-Base	MRPC	44.2 GB	(3.76 Gbps, 0.8 ms)	89.8%	89.1%	0.7%	✓	✓	✗	R	165.09 s
		RTE			54.5%	54.5%	0%					
		SST-2			89.5%	90.7%	-1.2%					
	GPT-2	-	44.0 GB		-	-	-					199.32 s
	BERT-Large	-	736.3 GB		-	-	-					1206.5 s
PrivTF [71]	BERT-Tiny	MRPC	9.5 GB	-	67%	66%	1%	✗	✗	✗	P	177 s
		MNLI			65%	65%	0%					
		QNLI			71%	70%	1%					
		SST-2			74%	73%	1%					
	BERT-Base	-	242 GB		-	-	-					2938 s
	TF-Base	-	45 GB		-	-	-					748 s
Donghwan et al. [66]	BERT	-	-	-	-	-	-	✓	✗	✓	P	20.86 s
ENSI [67]	Bitnet b1.58-3B	PIQA	-	-	71.65%	65.89%	5.76%	✓	✗	✓	P	-
		COPA			72.12%	71.26%	1.06%					-
		SST			56.57%	55.08%	1.49%					-
MERGE [87]	LLaMA-3-700M	-	-		-	-	-					956.08 s §
	GPT-2	Multiwoz	71.76 GB	3.76 Gbps, 0.1ms	-	-	-	✓	✓	✓	R	294.24 s
MARILL [104]	open-llama3b-v2	MTBench	-	(16 Gbps, 0.1 ms)	-	-	-	✗	✗	✓	P	-
		ParroT	-		-	-	-					-
	deepseek-coder-1.3B	MagiCoder	-	-	-	-	-					-
Curl [76]	BERT-Tiny	QNLI	1.34 GB	-	81.4%	80.3%	1.1%	✓	✗	✓	P	3.55 s
	BERT-Base	-	2.8 GB	-	-	-	-					13.63 s
	BERT-Large	-	5.66 GB	(3.76 Gbps, 0.1 ms)	-	-	-					33.93 s
	GPT-2	-	3.77 GB	-	-	-	-					16.61 s
	GPT-Neo	-	14.9 GB	-	-	-	-					103.4 s
Ditto[16]	BERT-Base	RTE	4.35 GB	(5 Gbps, 0.4 ms)	68.59%	67.87%	0.72%	✓	✓	✗	R	170.44 s
		CoLA			57.06%	54.17%	2.89%					
		QQP			87.96%	87.15%	0.81%					
		QNLI			91.62%	91.74%	-0.11%					
	GPT2-Base	Wikitext-103	8.25 GB		87.75%	87.01%	0.74%					51.32 s
Song et al. [102]	BERT-Tiny	MNLI	-	-	-	-	-	✗	✗	✓	P	0.19 s
	BERT-Base	QQP	-	-	-	-	-					0.75 s

Note: "✓" and "✗" indicate whether the code and executable are open-source and available, respectively. "✓" indicates that the paper mentions open-sourcing, but the code in the repository is still under development or coming soon. In the **Impl.** column, "P" indicates the data is taken directly from the original paper, and "R" means the result is reproduced in this work. "§": denotes the runtime is from the single layer. ">" indicates a crash during program execution.

Table 11: Resource requirements, tasks performed, dataset performance, and running time (part III).

Method	Model	Dataset	Comm.	Comm. Set.	Performance			Accessibility			Impl.	Time
					Plain	Enc.	Loss ↓	Code	Exe.	w/.GPU		
PermLLM. [99]	Single-Transformer		0.49 MB	(1 Gbps, 10 ms)	-	-	-	✗	✗	✓	P	0.12 s
				(100 Mbps, 20 ms)	-	-	-					0.24 s
	ChatGLM-6B		1.64 GB	(1 Gbps, 10 ms)	-	-	-					76 s
				(100 Mbps, 20 ms)	-	-	-					191 s
Truncformer[73]	LLaMA-7B	HumanEval	-	-	-	-	-	✗	✗	✗	P	-
	Gemma-2B	MBPP	-	-	-	-	-					-
Cipherformer [91]	Transformer	AG_News	420.4 MB		90.57%	87.70%	2.87%	✗	✗	✗	P	29.44 s †
		IMDB			83.46%	75.25%	8.21%					
		Yelp			91.73%	91.80%	0.02%					
FastQuery [105]	LLaMA-13B	-	-	-	-	-	-	✗	✗	✗	P	1.3× [84]
	LLaMA-30B	-			-	-	-					20.2×[84]
PolyTransformer [54]	BERT-Like	-	-	-	-	-	-	✗	✗	✓	P	211 s
Leporati et al. [56]	BERT-Tiny	SST-2	-	-	83.7%	79.0%	4.7%	✓	✓	✗	R	227 s
PrivCirNet [93]	ViT	CIFAR-10	-	-	93.54%	93.44%	0.10%	✓	✗	✗	P	-
		CIFAR-100			74.77%	74.71%	0.06%					
		Tiny-ImageNet			62.65%	61.79%	0.86%					
SecBERT [70]	BERT-Base	SST-2	-	-	83.7%	79.0%	4.7%	✗	✗	✗	P	2047 s
Nimbus[98]	BERT-Base	SST-2	-	(3 Gbps, 1 ms)	92.88%	92.66%	0.22%	✓	✓	✗	R	384.46 s
		QQP			91.22%	90.64%	0.58%					
		MNLI			84.74%	84.09%	0.65%					
		QNLI			91.28%	90.05%	1.23%					
		RTE			67.87%	66.79%	1.08%					
BOLT[33]	BERT-Base	MRPC	72.21 GB	(3 Gbps, 0.8 ms)	90.00%	90.53%	-0.53%	✓	✓	✗	R	369.19 s
		RTE			69.70%	69.68%	0.02%					
		SST-2			92.36%	91.74%	0.62%					
		SST-B			89.62%	87.97%	1.65%					
SecGPT [72]	BERT-Base	SST-2	-	-	83.7%	79.0%	4.7%	✗	✗	✗	P	64.78 s
SIGMA[50]	BERT-Tiny	SST-2	0.02 GB	(9.4 Gbps, 0.05 ms)	82.45%	82.57%	-0.12%	✓	✓	✓	R	1.36 s
		MRPC			71.07%	70.34%	0.73%					
		QNLI			81.42%	81.93%	-0.51%					
	BERT-Base	SST-2	0.99 GB		92.55%	92.55%	0.99%					13.99 s
		MRPC			84.31%	87.25%	-2.94%					
		QNLI			91.60%	91.63%	-0.03%					
	BERT-Large	SST-2	2.64 GB		93.58%	93.35%	0.23%					32.91 s
		MRPC			87.99%	88.48%	-0.49%					
		QNLI			92.23%	92.26%	-0.03%					
	GPT2-Base	Lambada	0.82 GB		32.46%	33.28%	-0.82%					13.16 s
	GPT-Neo	Lambada	4.03 GB		57.46%	57.81%	-0.35%					-
	LLaMA-2-7B	Lambada	12.07 GB		70.17%	69.92%	0.25%					-
	LLaMA-2-13B	Lambada	18.90 GB		73.14%	72.99%	0.15%					-
PPTIF [82]	Transformer	WMT	37.2 GB	-	-	-	-	✗	✗	✗	P	918 s
Power-Softmax [55]	GPT-1.4B	Pile	-	-	-	-	-	✗	✗	✗	P	-
	GPT-70M	Pile	-		-	-	-					400 s
	RoBERTa-Base	Pile	-		-	-	-					-
SecureTLM [69]	Transformer	-	219.7 MB	(3.76 Gbps, 0.1 ms)	-	-	-	✗	✗	✗	P	62.6 s
PriViT[101]	ViT	Tiny-ImageNet	-	-	-	64.73%	-	✓	✗	✗	P	-
Liu et al. [95]	BERT-Tiny	MRPC	1814 MB		76%	73%	3%	✓	✗	✓	P	13.89 s
		SST-2			83%	79%	4%					
		QNLI			82%	79%	3%					
	BERT-Medium	MRPC	-	-	89%	84%	5%					-
		SST-2			91%	88%	3%					
		QNLI			90%	85%	4%					
	BERT-Base	MRPC	-		92%	84%	8%					-
		SST-2			93%	89%	4%					
		QNLI			92%	89%	3%					
MpcFormer [15]	BERT-Base	MRPC	21.04 GB	(3.76 Gbps, 0.1 ms)	90.3%	88.7%	1.6%	✓	✓	✗	R	62.69 s
		RTE			69.7%	64.9%	4.8%					
		SST-2			93.1%	92.0%	1.1%					
		STS-B			89.1%	80.3%	8.8%					
		QNLI			91.7%	90.6%	1.1%					
SAL-VIT [80]	CCT	CIFAR-10	-	-	95.56%	95.92%	-0.36%	✗	✗	✓	P	36.47 s
		CIFAR-100			77.36%	77.86%	-0.5%					36.14 s
		Tiny-ImageNet			61.60%	63.49%	-1.89%					63.88 s
Wang et al. [18]	Transformer	-	-	-	-	-	-	✗	✗	✓	P	-
MPC-VIT [81]	ViT	CIFAR-10	-	-	-	93.38%	-	✓	✓	✓	R	5.57 s
		CIFAR-100			-	75.38%	-					-

Note: "✓" and "✗" indicate whether the code and executable are open-source and available, respectively. In the **Impl.** column, "P" indicates the data is taken directly from the original paper, and "R" means the result is reproduced in this work. "†" indicates that the value is the sum of both the online and offline phases. "×[84]" indicates that the value is x times lower than that of Bumblebee [84].

Table 12: Resource requirements, tasks performed, dataset performance, and running time (part IV).

Method	Model	Dataset	Comm.	Comm. Set.	Performance			Accessibility			Impl.	Time
					Plain	Enc.	Loss ↓	Code	Exe.	w/.GPU		
RNA-VIT [78]	CCT	CIFAR-10	-	-	95.56%	94.97%	0.59%	✗	✗	✓	P	14.04 s
		CIFAR-100			77.36%	79.04%	-1.68%					14.13 s
		Tiny-ImageNet			61.60%	65.86%	-4.26%					44.27 s
CipherGPT [90]	GPT-2	WikiText-103	14.74 GB	-	-	99.22%	-	✗	✗	✗	P	1180.9 s
Privformer [13]	Transformer	-	1142.56 MB	(4.93 Gbps, 1.17 ms)	-	-	-	✗	✗	✗	P	31.91 s
		-		(97.4 Mbps, 141.67 ms)	-	-	-					656.0 s
Primer [52]	BERT-Base	MNLI-m	3.6 GB	(800 Mbps, 2.3 ms)	-	77.6%	-	✗	✗	✓	P	434.8 s †
		MRPC			-	79.3%	-					
		SST-2			-	88.2%	-					
		SQuAD1			-	86.2%	-					
		SQuAD2			-	76.6%	-					
	BERT-Small	MNLI-m	1.8 GB		-	81.6%	-					364.1 s †
		MRPC			-	84.5%	-					
		SST-2			-	91.1%	-					
		SQuAD1			-	88.5%	-					
		SQuAD2			-	78.2%	-					
	BERT-Base	MNLI-m	3.6 GB		-	84.6%	-					434.8 s †
		MRPC			-	86.3%	-					
		SST-2			-	92.5%	-					
		SQuAD1			-	90.7%	-					
		SQuAD2			-	80.3%	-					
	BERT-Medium	MNLI-m	3.9 GB		-	85.4%	-					497.9 s†
		MRPC			-	86.3%	-					
		SST-2			-	93.1%	-					
		SQuAD1			-	92.2%	-					
		SQuAD2			-	81.6%	-					
	BERT-Large	MNLI-m	7.9 GB		-	86.6%	-					678 s †
		MRPC			-	87.6%	-					
		SST-2			-	93.5%	-					
		SQuAD1			-	93.1%	-					
		SQuAD2			-	82.9%	-					
Iron [89]	BERT-Tiny	MNLI	4.0 GB	(377 Mbps, 0.8 ms)	69.75%	68.90%	0.85%	✗	✗	✗	P	43.2 s
		MRPC			76.90%	76.88%	0.02%					
		QNLI			74.94%	75.05%	0.11%					
		SST-2			71.62%	71.89%	0.27%					
	BERT-Medium	MNLI	28.6 GB		-	-	-					366 s
		MRPC			-	-	-					
		QNLI			-	-	-					
		SST-2			-	-	-					
	BERT-Base	MNLI	177 GB		-	-	-					266 min
		MRPC			-	-	-					
		QNLI			-	-	-					
		SST-2			-	-	-					
	BERT-Large	MNLI	640 GB		-	-	-					961 min
		MRPC			-	-	-					
		QNLI			-	-	-					
		SST-2			-	-	-					
THE-X [12]	BERT-Tiny	SST-2	-	-	82.45%	82.11%	0.34%	✗	✗	✗	P	4700 s
		MRPC			70.10%	68.87%	1.13%					
		STS-B			73.66%	70.97%	2.69%					
		QQP			84.03%	83.63%	0.67%					
		MNLI			70.75%	69.08%	1.42%					
		QNLI			78.38%	78.20%	0.18%					
		RTE			58.56%	58.12%	0.44%					

Note: "✓" and "✗" indicate whether the code and executable are open-source and available, respectively. In the **Impl.** column, "P" indicates the data is taken directly from the original paper, and "R" means the result is reproduced in this work. "†" indicates that the value is the sum of both the online and offline phases.

Table 13: Representative Transformer Models, Their Categories, Parameters, and Dimensions (M and B denote million and billion parameters, respectively).

Category	Model	Params	Layers	d_{model}	Heads	d_{head}	Q/K/V dims	FFN dims
Encoder-only	BERT-Tiny	4M	2	128	2	64	$128 \to 128$	$128 \to 512 \to 128$
	BERT-Base	110M	12	768	12	64	$768 \to 768$	$768 \to 3072 \to 768$
	BERT-Large	340M	24	1024	16	64	$1024 \to 1024$	$1024 \to 4096 \to 1024$
	RoBERTa-Base	125M	12	768	12	64	$768 \to 768$	$768 \to 3072 \to 768$
	RoBERTa-Large	355M	24	1024	16	64	$1024 \to 1024$	$1024 \to 4096 \to 1024$
	ChatGLM2-6B	6.2B	28	4096	32	128	$4096 \to 4096$	$4096 \to 16384 \to 4096$
Decoder-only	GPT-2 Small	117M	12	768	12	64	$768 \to 768$	$768 \to 3072 \to 768$
	GPT-2 Medium	345M	24	1024	16	64	$1024 \to 1024$	$1024 \to 4096 \to 1024$
	GPT-2 Large	774M	36	1280	20	64	$1280 \to 1280$	$1280 \to 5120 \to 1280$
	GPT-Neo	1.3B	24	2048	16	128	$2048 \to 2048$	$2048 \to 8192 \to 2048$
	LLaMA-3B	3B	26	3200	32	100	$3200 \to 3200$	$3200 \to 8640 \to 3200$
	LLaMA-7B	7B	32	4096	32	128	$4096 \to 4096$	$4096 \to 11008 \to 4096$
	LLaMA-13B	13B	40	5120	40	128	$5120 \to 5120$	$5120 \to 13824 \to 5120$
Vision Transformer	ViT-Tiny/16	5.7M	12	192	3	64	$192 \to 192$	$192 \to 768 \to 192$
	ViT-Base/16	86M	12	768	12	64	$768 \to 768$	$768 \to 3072 \to 768$
	ViT-Large/16	307M	24	1024	16	64	$1024 \to 1024$	$1024 \to 4096 \to 1024$