A Survey on Private Transformer Inference with Homomorphic Encryption

Yang Li[a], **Xinyu Zhou**[a], **Yitong Wang**[a], **Liangxin Qian**[a], **Jun Zhao**[a]

[a] *Nanyang Technological University* {yang048, xinyu003, yitong002, qian0080}@e.ntu.edu.sg, junzhao@ntu.edu.sg

1. Introduction

Transformer models have revolutionized the field of AI, powering applications like OpenAI ChatGPT and Microsoft Bing. Nonetheless, their current deployment still raises privacy concerns: clients must upload their sensitive user data to the server for inference service. This reliance raises concerns over data misuse, like unauthorized processing, indefinite storage, or resale to third parties. Private Transformer Inference (PTI) addresses this by leveraging cryptographic techniques. It enables model inference without the server learning the user's input or the user learning anything about the server's model, except for the inference results. Specifically, this paper reviews recent PTI advancements (2022–2024) focusing on Homomorphic Encryption (HE). To our best knowledge, surveys focusing on private transformer inference with HE do not exist so far. This paper aims to give a brief overview of PTI with HE and its implementations.

2. A brief introduction to HE

HE enables computations on encrypted data, yielding results identical to plaintext operations after decryption. It uses a public key for encryption and a secret key for decryption. The HE schemes can be further categorized by the operations in the circuit and its computational depth. Due to space limitations, further details are omitted. Notably, current HE schemes only support linear operations, i.e., additions and multiplications.

3. Taxonomy of transformer layers in cryptographic contexts

Transformer layers could be classified as linear (embedding, attention matrix multiplication, feed-forward) and non-linear (Softmax, GELU, LayerNorm). Fig. 1 illustrates the basic transformer encoder architecture.

3.1 Linear layers

Linear layers consist of matrix multiplications, which are compatible with HE but inefficient with naive implementations [1]. Two mainstream methods address this:

1) Encoding plaintexts into SIMD slots.

2) Encoding plaintexts as polynomial coefficients.

The SIMD technique batches multiple elements into one ciphertext, enabling parallel computation to reduce amortized costs. Studies [2, 3, 4, 5] have leveraged SIMD in their implementations. However, when applied to MatMul, SIMD requires expensive homomorphic rotations to perform the summation. To mitigate this, [2] uses the Baby-Step-Giant-Step (BSGS) method to reduce rotations, while [3] introduces a slots folding approach. In another way, studies [6, 7] show that encoding plaintexts

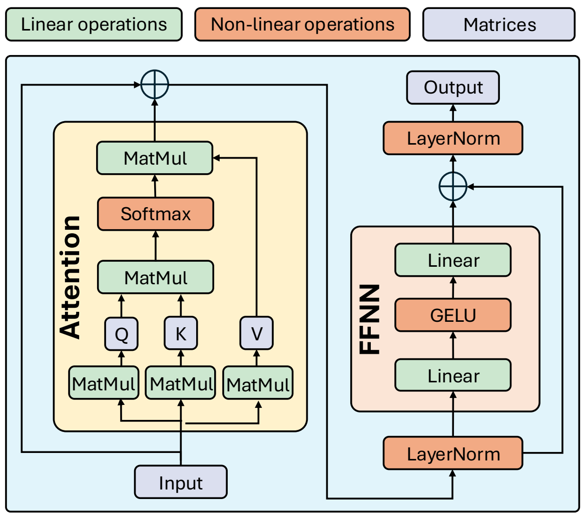

Fig. 1: The architecture of a transformer encoder

into polynomial coefficients can directly compute the dot product, eliminating rotations. Furthermore, [8] proposes a compact encoding method to address sparsity and reduce communication, and [9] customizes a Vector Oblivious Linear Evaluation-based protocol in GPT, lowering the amortized cost of auto-regressive response generation.

3.2 Non-linear layers

Securing non-linear layers is challenging due to their cryptographic complexity. Those layers mainly include Softmax, GELU and LayerNorm.

Softmax. The bottleneck in Softmax is to efficiently calculate the underlying $\exp(x)$. Some studies employ aggressive crypto-friendly functions, e.g., $(x + c)^2$, to directly replace the $\exp(x)$ in Softmax. However, such an approach often brings significant accuracy drops and requires knowledge distillation. In contrast, [4] approximates $\exp(x)$ using a 6-degree Maclaurin series for accuracy. Similarly, studies [8, 3] design piecewise polynomials with Taylor Series for approximation.

GELU. The nonlinearity in GELU comes from the Gaussian error function $\mathrm{erf}(x)$. Studies [1, 11] directly replace GELU with crypto-friendly RELU (i.e., $\max(0, x)$) since support for comparison operations in cryptography is relatively well established. Besides, since GELU is almost linear with a larger or smaller input, studies [8, 2, 3] suggest an efficient low-degree polynomial (e.g., $n \leq 4$ in [2]) for approximation within the short interval around 0.

LayerNorm. The difficulty in LayerNorm is the required reciprocal square root operation $1/\sqrt{x}$. Studies [3, 11] employ Newton-like methods to compute $1/\sqrt{x}$ iteratively. Other studies avoid non-linear computations by altering the architecture of LayerNorm: [1] directly removes the computation of mean and standard deviation,

Table 1: Resource requirements, tasks performed, dataset performance, and runtime.

Study	Techniques	Model	Dataset	Comm.	Comm. Set.	Performance			Runtime
						Plain	Enc.	Loss ↓	
[7]	HE+MPC	BERT-Base	SST-2	280.99 GB	(3 Gbps, 0.8 ms)	92.36 %	92.77 %	-0.41 %	475 s
[8]	HE+MPC	BERT-Base	QNLI	-	(1 Gbps, 0.5 ms)	90.30 %	90.20 %	0.10 %	-
		BERT-Large	-	20.85 GB		-	-	-	404.4 s
		LLaMA-7B	-	6.82 GB		-	-	-	832.2 s
		ViT-Base	ImageNet	14.44 GB		89.44 %	89.13 %	0.31 %	234 s
[2]	HE+MPC	BERT-Base	SST-2	25.74 GB	(3 Gbps, 0.8 ms)	92.36 %	92.78 %	-0.42 %	185 s
[1]	HE	BERT-Tiny	SST-2	-	-	82.45 %	82.11 %	0.34 %	≈ 4700 s
[4]	HE	BERT-Tiny	SST-2	-	-	83.7 %	79.0 %	4.7 %	214 s
[5]	HE	BERT-Base	MRPC	-	-	85.29 %	84.80 %	0.49 %	625.8 s
[10]	HE	Roberta-Base	SST-2	-	-	94.80 %	93.35 %	1.45 %	≈ 400 s
[3]	HE	BERT-Base	SST-2	0.16 GB	(100 Mbps, 80 ms)	92.36 %	92.11 %	0.25 %	857 s
		LLaMA-3B	SST-2			94.94 %	94.46 %	0.48 %	1088 s

leaving them achieved by learnable affine parameters. [4] precomputes the values of mean and standard deviation to simplify the computation.

4. Reported Experiments

Table 1 shows reported experiments in selected studies. Notably, some studies [7, 8, 2] employ a mixed method of "HE+MPC". They use HE for linear layers and secure Multi-Party Computation (MPC) for non-linear ones. Other studies only use HE to design the whole circuit.

Runtime. Studies using "HE+MPC" often have a better runtime performance at the cost of significant communication overhead. For example, [2] could evaluate the BERT-Base model in 185 s with 25.74 GB overhead. HE-only solutions require the longest runtime. Latest studies [10, 5, 3] all require over 400 s for evaluation on the BERT-Base model. Even for the smaller BERT-Tiny model, [4] still requires over 200 s. In particular, the most consuming part of those studies is the *Bootstrapping* operation, which is often used to "refresh" a ciphertext to reduce noise. The bootstrapping part in [3] and [5] accounts for 37.72% and 53.96% of the total runtime, respectively.

Accuracy. Linear computations are often well supported by HE. Hence, most of the accuracy drop in Table 1 comes from the treatments of non-linear layers. For instance, [4] employed aggressive substitutions for the LayerNorm functions, leading to an accuracy drop of 4.7% on the STS-2 dataset. In contrast, other studies usually designed approximations more carefully or utilized knowledge distillation for re-training, so the accuracy drop is limited.

References

[1] 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. *arXiv preprint arXiv:2206.00216*, 2022.

[2] Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. Bolt: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 130–130. IEEE Computer Society, 2024.

[3] Jiawen Zhang, Jian Liu, Xinpeng Yang, Yinghao Wang, Kejia Chen, Xiaoyang Hou, Kui Ren, and Xiaohu Yang. Secure transformer inference made non-interactive. *Cryptology ePrint Archive*, 2024.

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

[5] Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. Thor: Secure transformer inference with homomorphic encryption. *Cryptology ePrint Archive*, 2024.

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

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

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

[9] 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*, 2023.

[10] 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. *arXiv preprint arXiv:2410.09457*, 2024.

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