Private LLM Inference with Homomorphic Encryption

Lawrence Lim
University of California, Santa Barbara
Santa Barbara, California, USA
lawrenceklim@ucsb.edu

Divyakant Agrawal
University of California, Santa Barbara
Santa Barbara, California, USA
divyagrawal@ucsb.edu

Amr El Abbadi
University of California, Santa Barbara
Santa Barbara, California, USA
elabbadi@ucsb.edu

Abstract

This tutorial introduces the emerging research area of private large language model (LLM) inference using Fully Homomorphic Encryption (FHE). It covers the basics of homomorphic encryption, key architectural details of transformer-based LLMs, and existing techniques for enabling end-to-end encrypted LLM inference. Finally, the tutorial concludes with open research questions and future directions in this rapidly evolving field.

Keywords

Private LLM Inference, Homomorphic Encryption

1 Introduction

As large language models (LLMs) are increasingly deployed across a wide range of applications, protecting the privacy of user queries has become a critical concern. *Fully homomorphic encryption* (FHE) offers a compelling approach to this problem by enabling computation directly over encrypted data, allowing an untrusted server to execute LLM inference without learning the underlying inputs. Despite its promise, private LLM inference under FHE presents substantial technical challenges, including efficient encrypted matrix multiplications, support for multi-head attention, approximation of nonlinear layers such as softmax, GELU, and layer normalization, and the mitigation of FHE's high computational overhead. Recent work has made meaningful progress toward addressing these challenges [12, 14, 20, 23, 25, 26, 28, 32], but existing systems remain far from practical deployment, leaving significant room for further research.

Tutorial overview: This 90-minute tutorial is delivered as a lecture and is designed to be accessible to researchers from both academia and industry, without requiring prior expertise in cryptography or machine learning. We will cover the basics of homomorphic encryption, focusing in particular on the CKKS scheme [9, 10]. We then explore the internal structure of transformer models and how those components, including matrix multiplications and nonlinear layers, can be implemented efficiently under homomorphic encryption. The tutorial culminates in a discussion of end-to-end private LLM inference systems, such as Thor [23], Tricycle [20], and Powerformer [25]. The tutorial is organized as follows:

(1) **Motivation and System Model (10 minutes)**
(2) **Homomorphic Encryption (10 minutes)**
(3) **Transformer Models (10 minutes)**
(4) **Matrix Multiplications in HE (25 minutes)**
(5) **Nonlinear Layers in HE (25 minutes)**
(6) **End-to-end Systems and Future Directions (10 minutes)**

EDBT '26, Tampere (Finland)

2 Motivation and System Model

Recently, large data management has evolved to improve and generalize standard database querying by incorporating LLMs into query processing. This allows users to expand the realm of their queries beyond the standard closed database models to include more general information. However, a significant barrier to the broader adoption of LLMs is the need to query them with sensitive or private data. For example, healthcare practitioners might want to query an LLM agent to summarize a patient's clinical history without revealing private information to the LLM provider. To enable private LLM inference, four categories of approaches are currently being explored:

- Local LLM inference
- Secure enclaves
- Multi-party computation (MPC)
- Fully homomorphic encryption (FHE)

In *local LLM inference*, users execute LLM models entirely on their own devices. This approach offers strong privacy guarantees, since no data leaves the user's machine. However, it has two major limitations: it assumes users have access to model weights, which may not be released by model providers, and it requires sufficient local computational resources, which many users may lack. In the *secure enclave* model, LLM inference is performed inside trusted execution environments to protect user inputs. While this approach can be practical, it relies on strong trust assumptions in enclave hardware and its security guarantees. Nevertheless, secure enclaves have seen industry adoption due to their relative ease of deployment, including by companies such as Duality Technologies[1], NEAR.AI[2], and Tinfoil[3]. *Multi-party computation (MPC)*–based approaches use a combination of MPC and homomorphic encryption to jointly compute LLM inference without revealing private inputs [19, 22, 24]. These systems must be carefully designed to avoid leaking privacy and typically incur significant communication overhead, making them challenging to scale. Finally, *fully homomorphic encryption*, the primary focus of this tutorial, enables LLM inference to be performed directly on encrypted data, offering strong privacy guarantees without requiring trust in the server [12, 20, 25, 26, 28, 32]. The main drawback of FHE-based approaches is their high computational cost. Despite this challenge, FHE has attracted interest from both academia and industry, including companies such as Cornami[4] and Zama[5].

3 Homomorphic Encryption

Homomorphic Encryption (HE) enables computations to be performed directly on encrypted data, without revealing the underlying plaintexts to the party performing the computation. This property makes HE particularly attractive for privacy-preserving

[1] https://dualitytech.com
[2] https://near.ai
[3] https://tinfoil.sh
[4] https://cornami.com/generative-ai/
[5] https://www.zama.org

10.48786/edbt.2026.78

machine learning, where a server can evaluate a model on sensitive user inputs without learning either the inputs or the outputs.

The primary HE scheme used in modern machine learning applications is CKKS [9, 10]. CKKS is an approximate homomorphic encryption scheme designed for real-valued arithmetic. To ensure security, ciphertexts inherently contain approximation noise which increases with homomorphic operations. Also, CKKS encrypts vectors of floating-point values and supports Single Instruction Multiple Data (SIMD) execution, enabling the same arithmetic operation to be applied in parallel across many packed values within a ciphertext. Effectively exploiting this SIMD parallelism is critical for achieving practical performance.

In addition to key generation, encryption, and decryption, CKKS supports the following homomorphic operations:

- **Add**: Element-wise addition between two ciphertexts or between a ciphertext and a plaintext.
- **Multiply**: Element-wise multiplication between two ciphertexts or between a ciphertext and a plaintext.
- **Rotate**: Cyclic shift of the packed vector slots within a ciphertext.

Applying CKKS to real-world applications requires expressing *all* computations as compositions of these primitive operations. While linear algebra operations such as matrix multiplication can be implemented efficiently, nonlinear layers commonly used in LLMs (such as softmax) must be approximated using polynomials. These polynomial approximations are often costly because they require significant multiplicative depth, which can trigger an expensive procedure known as *bootstrapping* [2, 4, 5, 7, 16]. This is because in CKKS, each ciphertext is associated with a *level*, which represents the remaining multiplicative depth budget. Every multiplication consumes one level. Once this budget is exhausted, further multiplications require bootstrapping, an extremely expensive operation that refreshes the ciphertext's budget. Because LLM inference consists of deep computational pipelines, bootstrapping frequently becomes the dominant performance bottleneck. After bootstrapping, the next most expensive operations are ciphertext–ciphertext multiplications and rotations, both of which involve costly key-switching operations. Consequently, HE system design is governed by two primary constraints:

(1) **Minimizing multiplicative depth** to reduce or eliminate the need for bootstrapping.
(2) **Minimizing expensive operations**, such as key-switching operations, by carefully packing data to maximize SIMD utilization and by designing algorithms that are well-suited to HE.

4 Transformer Models

An LLM is a neural network architecture composed of an embedding layer, a series of transformer layers, and output layers, as illustrated in Figure 1 and described in [18, 29]. Given an input string, the text is first tokenized into a sequence of discrete tokens. These tokens are then mapped to continuous vector representations via an embedding layer, producing an input matrix to the transformer. Existing work on private LLM inference assumes that tokenization and embedding are performed locally on the client device before encryption [20, 23, 25]. Accordingly, we do not focus on the embedding layer in detail.

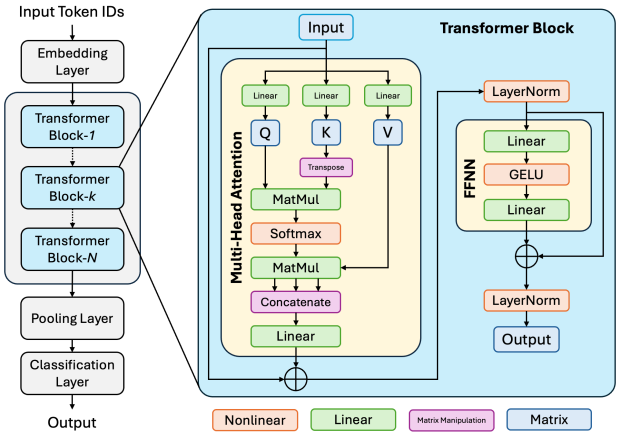

Figure 1: An example transformer model such as BERT-Base [13].

To enable private inference with FHE, every computation within a transformer layer must be carried out directly on ciphertexts. A transformer layer takes a matrix as input and produces a matrix of the same dimension as output, where each row corresponds to the embedding of a token in the sequence. Each transformer layer consists of two primary components: *multi-head attention* and a *feed-forward neural network*. In multi-head attention, the attention mechanism enables the model to capture relationships between tokens in the input sequence, while the feed-forward network is a set of row-wise fully connected layers.

At the core of attention is a weighted aggregation over the input tokens, where the weights reflect the relevance of one token to another. This operation is formalized as

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{1}$$

where Q, K, and V are the query, key, and value matrices, and d_k denotes the dimensionality of the key vectors. The matrices Q, K, and V are obtained by multiplying the input matrix to the transformer layer with learned weight matrices W_Q, W_K, and W_V, respectively.

In practice, transformers employ *multi-head attention*. Rather than computing a single attention operation, the input is projected into multiple independent (Q_i, K_i, V_i) triples. Attention is computed independently for each head, allowing the model to capture different types of relationships in parallel. The outputs of all heads are then concatenated and then multiplied with an attention output projection weight matrix W_o, which mixes information across heads and maps the concatenated representation back to the model dimension.

Following the attention computation, each transformer layer applies a residual connection that adds the attention output to the original layer input. This is followed by a row-wise layer normalization operation, defined as

$$\text{Layer Normalization}(x) = \gamma \cdot \frac{x - \mu}{\sigma} + \beta, \tag{2}$$

where μ and σ are the mean and standard deviations of each row and γ and β are weights.

After the first layer normalization, the second component, the feed-forward neural network (FFNN), is applied. The FFNN is defined as

$$\text{FFNN}(x) = W_{\text{ff2}} \cdot \text{GELU}(W_{\text{ff1}}x + b_{\text{ff1}}) + b_{\text{ff2}}. \tag{3}$$

where W_{ff1} and W_{ff2} are weight matrices, b_{ff1} and b_{ff2} are biases, $\text{GELU}(x) = x \cdot \Phi(x)$ is the Gaussian Error Linear Unit nonlinear layer, and $\Phi(x)$ denotes the cumulative distribution function of the standard normal distribution. As with attention, the FFNN output is combined with a residual connection, followed by a second layer normalization, before being passed to the next transformer layer or output layers.

After all transformer layers, the model applies one or more output layers, depending on the task. This may include pooling operations that extract a single embedding (e.g., corresponding to a specific token), and additional linear and nonlinear layers. For text generation models, the final hidden states are typically projected through a linear layer to produce logits over the vocabulary, after which a temperature-scaled softmax is applied and a token is sampled from the resulting distribution.

5 Matrix Multiplications in HE

Matrix multiplications account for a substantial fraction of the computation in transformer inference. Ciphertext–plaintext multiplications arise when applying model weight matrices, while ciphertext–ciphertext multiplications are required for computing attention. Hence, supporting both ciphertext–plaintext and ciphertext–ciphertext matrix multiplications is essential.

Since CKKS operates in a SIMD manner over vectors of encrypted values, matrices must first be flattened into vectors, referred to as *packing*. Matrix multiplications are then implemented using sequences of element-wise multiplications, element-wise additions, and rotations over these packings. To achieve high performance, a packing should make efficient use of ciphertext slots and matrix multiplications should maximize SIMD parallelism and minimize both multiplicative depth and expensive key-switching operations.

The research literature broadly adopts two system-level approaches that differ in how packings are constructed. One approach batches multiple independent inputs from different user queries into the same ciphertext (e.g., NEXUS [31], MOAI [32], and ARION [30]). This strategy can significantly reduce the number of required rotations, leading to improved amortized performance. However, it relies on strong assumptions, such as several users sharing a cryptographic key or a single user submitting several queries simultaneously.

The alternative approach avoids batching across inputs and instead focuses on efficient single-input execution (e.g., Thor [23], Tricycle [20], Powerformer [25], Rovida *et al.* [28], and Garimella *et al.* [14]). A representative packing scheme in this category is the *bicyclic encoding* [6], which packs a matrix into a vector by traversing its diagonals. For example, consider the matrix A:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

The bicyclic encoding of A is $(1, 6, 11, 4, 5, 10, 3, 8, 9, 2, 7, 12)$. When two matrices are encoded in this form, their matrix product can be computed as a sum of element-wise products over rotated ciphertexts, achieving optimal multiplicative depth of one.

Although bicyclic encodings are expressive enough to support all matrix multiplications in a transformer, applying them naively can lead to suboptimal performance. As a result, several important optimizations have been proposed in the broader literature to improve the packings and matrix multiplications:

- **Multi-head packing:** Packing multiple attention heads into a single ciphertext enables parallel evaluation of independent matrix multiplications and better utilization of SIMD. Tricycle [20], for example, generalizes bicyclic encodings for three-dimensional tensors to support this parallelism.
- **Baby-Step Giant-Step (BSGS):** BSGS [8] reduces the number of rotations required when summing several rotated ciphertexts.
- **Hoisting and double hoisting:** These techniques [3, 15] amortize the cost of multiple rotations by computing the shared intermediate components only once.
- **Lazy relinearization:** Deferring relinearization [23] reduces overhead during ciphertext–ciphertext matrix multiplications.
- **Complexification:** By exploiting CKKS's native support for complex numbers, values can be packed into both the real and imaginary components of ciphertext slots, effectively doubling packing density [23, 25].

While different systems adopt distinct packing strategies and combinations of optimizations, most practical HE-based transformer inference frameworks rely on some subset of these techniques to achieve efficient matrix multiplication.

6 Nonlinear Layers in HE

Homomorphic encryption schemes such as CKKS natively support only a limited set of operations—addition, multiplication, and rotations. As a result, nonlinear functions commonly used in neural networks must be implemented using polynomial approximations. In transformer models, the primary nonlinearities that require approximation are softmax, layer normalization, and GELU.

Existing work on private LLM inference generally follows one of two strategies. The first strategy replaces standard nonlinear functions with HE-friendly alternatives that are cheaper to evaluate under encryption [25, 26]. For example, some or all of softmax or layer normalization may be substituted with simplified linear variants. Although this approach can significantly reduce the computation cost, it requires retraining the model to accommodate the modified nonlinearities, which limits compatibility with existing pretrained models.

The second strategy is to faithfully approximate the original nonlinear functions used in standard transformers [20, 23, 30, 31]. This is typically done using polynomial approximation techniques such as minimax, Chebyshev, or Taylor series expansions. In some cases, specialized methods, such as Newton's iteration, can be used to approximate inverse functions. In general, higher approximation accuracy requires higher-degree polynomials, which in turn increases multiplicative depth and overall computational cost.

As a concrete example [20, 30, 31], consider the softmax function, which applies an exponentiation followed by a normalization:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}. \tag{4}$$

Approximating softmax under HE therefore requires approximating both exponentiation and division.

Exponentiation can be approximated using Chebyshev or minimax polynomials, or via the identity $e^x \approx \left(1 + \frac{x}{2^k}\right)^{2^k}$, for a

constant k (typically $k \in 7, 8$). This reduces exponentiation to a sequence of squaring operations. Similarly, the reciprocal function can be approximated using polynomial methods or with Newton's method. Given an initial approximation $x_0 \approx 1/a$, Newton's method can iteratively refine the estimate as $x_{k+1} = x_k \cdot (2 - ax_k)$. Each iteration improves accuracy at the cost of additional multiplicative depth.

Beyond approximation cost, softmax represents a numerical stability challenge under CKKS. Exponentiation amplifies large positive inputs while driving large negative inputs toward zero, which can exacerbate CKKS noise growth and cause small values to be subsumed by encryption noise. This issue is mitigated using techniques such as subtracting by a constant offset [31], subtracting by an estimated maximum value [20, 30], or applying iterative normalization procedures [11, 23].

7 End-to-end Systems and Future Directions

Recent systems have made substantial progress toward practical private LLM inference under homomorphic encryption. State-of-the-art GPU-based implementations can evaluate a BERT-Base model in 602 seconds using faithful approximations (Thor [23]), or in 344 seconds by replacing standard components with more HE-friendly alternatives and retraining the model accordingly (Powerformer [25]). CPU-based Tricycle [20] evaluates a BERT-Tiny [17] model with faithful approximations in 322 seconds. Input batching techniques in GPU-based MOAI [32] achieve amortized inference latency of 284 seconds and the very recent work of CPU-based ARION [30] achieves amortized latency of 225 seconds. Across these systems, empirical results demonstrate that polynomial approximations introduce only minimal end-to-end accuracy degradation.

Despite this progress, fully end-to-end private text generation under HE remains an open challenge. Moving beyond private inference for a single token, efficient autoregressive generation requires a private key–value (KV) cache to reuse past attention computations across decoding steps. In a fully non-interactive setting, such a private KV cache must be integrated with recent advances in HE-based token sampling [1, 21, 27], as well as an embedding layer implemented entirely under HE. Realizing these components within a single system will require new algorithmic techniques that further reduce computational cost and improve scalability.

8 Biography

Lawrence Lim is a Ph.D. candidate at UC Santa Barbara advised by Professors Divyakant Agrawal and Amr El Abbadi. He was the primary developer behind the private LLM inference system Tricycle [20] and token sampling algorithm Hyperion [21].

Divyakant Agrawal is a Distinguished Professor of Computer Science at the University of California at Santa Barbara. Over the course of his career, he has published more than 400 research articles and has mentored approximately 50 Ph.D. students. He served as a journal editor for several database journals and is currently serving as the Chair of ACM Special Interest Group on Management of Data (SIGMOD). He is a Fellow of the ACM, the IEEE, and the AAAS.

Amr El Abbadi is a Distinguished Professor of Computer Science at the University of California, Santa Barbara. Prof. El Abbadi is an ACM Fellow, AAAS Fellow, and IEEE Fellow. He has served as a journal editor for several database journals and has been Program Chair for multiple databases and distributed systems conferences. He has published over 400 articles in databases and distributed systems and has supervised over 40 Ph.D. students.

Acknowledgments

This work was supported in part by a Google PSS Privacy Research Faculty Award. The authors thank Google for its generous support.

References

[1] Matan Avitan, Moran Baruch, Nir Drucker, Itamar Zimerman, and Yoav Goldberg. 2025. Efficient Decoding Methods for Language Models on Encrypted Data. arXiv:2509.08383 [cs.LG] https://arxiv.org/abs/2509.08383

[2] Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. 2021. Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-sparse Keys. In *Advances in Cryptology – EUROCRYPT 2021*, Anne Canteaut and François-Xavier Standaert (Eds.). Springer International Publishing, Cham, 587–617.

[3] Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. 2021. Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-sparse Keys. In *Advances in Cryptology – EUROCRYPT 2021*, Anne Canteaut and François-Xavier Standaert (Eds.). Springer International Publishing, Cham, 587–617.

[4] Jean-Philippe Bossuat, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. 2022. Bootstrapping for Approximate Homomorphic Encryption with Negligible Failure-Probability by Using Sparse-Secret Encapsulation. In *Applied Cryptography and Network Security*, Giuseppe Ateniese and Daniele Venturi (Eds.). Springer International Publishing, Cham, 521–541.

[5] Hao Chen, Ilaria Chillotti, and Yongsoo Song. 2019. Improved Bootstrapping for Approximate Homomorphic Encryption. In *Advances in Cryptology – EUROCRYPT 2019*, Yuval Ishai and Vincent Rijmen (Eds.). Springer International Publishing, Cham, 34–54.

[6] Jingwei Chen, Linhan Yang, Wenyuan Wu, Yang Liu, and Yong Feng. 2025. Homomorphic Matrix Operations Under Bicyclic Encoding. *IEEE Transactions on Information Forensics and Security* 20 (2025), 1390–1404. doi:10.1109/TIFS.2024.3490862

[7] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. 2018. Bootstrapping for Approximate Homomorphic Encryption. In *Advances in Cryptology – EUROCRYPT 2018*, Jesper Buus Nielsen and Vincent Rijmen (Eds.). Springer International Publishing, Cham, 360–384.

[8] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. 2018. Bootstrapping for Approximate Homomorphic Encryption. In *Advances in Cryptology – EUROCRYPT 2018*, Jesper Buus Nielsen and Vincent Rijmen (Eds.). Springer International Publishing, Cham, 360–384.

[9] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. 2018. A full RNS variant of approximate homomorphic encryption. In *International Conference on Selected Areas in Cryptography*. Springer, 347–368.

[10] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Advances in Cryptology – ASIACRYPT 2017*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer International Publishing, Cham, 409–437.

[11] Wonhee Cho, Guillaume Hanrot, Taeseong Kim, Minje Park, and Damien Stehlé. 2024. Fast and Accurate Homomorphic Softmax Evaluation. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security* (Salt Lake City, UT, USA) *(CCS '24)*. Association for Computing Machinery, New York, NY, USA, 4391–4404. doi:10.1145/3658644.3670369

[12] Leo De Castro, Daniel Escudero, Adya Agrawal, Antigoni Polychroniadou, and Manuela Veloso. 2025. EncryptedLLM: Privacy-Preserving Large Language Model Inference via GPU-Accelerated Fully Homomorphic Encryption. In *Proceedings of the 42nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 267)*, Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (Eds.). PMLR, 12677–12688. https://proceedings.mlr.press/v267/de-castro25a.html

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[14] Karthik Garimella, Negar Neda, Austin Ebel, Nandan Kumar Jha, and Brandon Reagen. 2025. Network and Compiler Optimizations for Efficient Linear Algebra Kernels in Private Transformer Inference (Invited Paper). In *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. 1–10. doi:10.1109/ICCAD66269.2025.11240733

[15] Shai Halevi and Victor Shoup. 2014. Algorithms in helib. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*. Springer, 554–571.

[16] Kyoohyung Han and Dohyeong Ki. 2020. Better Bootstrapping for Approximate Homomorphic Encryption. In *Topics in Cryptology – CT-RSA 2020*, Stanislaw Jarecki (Ed.). Springer International Publishing, Cham, 364–390.

[17] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. arXiv:1909.10351 [cs.CL] https://arxiv.org/abs/1909.10351

[18] Yang Li, Xinyu Zhou, Yitong Wang, Liangxin Qian, and Jun Zhao. 2024. A Survey on Private Transformer Inference. arXiv:2412.08145 [cs.CR] https://arxiv.org/abs/2412.08145

[19] Zhengyi Li, Kang Yang, Jin Tan, Wen-jie Lu, Haoqi Wu, Xiao Wang, Yu Yu, Derun Zhao, Yancheng Zheng, Minyi Guo, et al. 2024. Nimbus: Secure and Efficient Two-Party Inference for Transformers. *Advances in Neural Information Processing Systems* 37 (2024), 21572–21600.

[20] Lawrence Lim, Vikas Kalagi, Divyakant Agrawal, and Amr El Abbadi. 2025. Tricycle: Private Transformer Inference with Tricyclic Encodings. Cryptology ePrint Archive, Paper 2025/1200. https://eprint.iacr.org/2025/1200

[21] Lawrence Lim, Jiaming Liu, Vikas Kalagi, Amr El Abbadi, and Divyakant Agrawal. 2025. Hyperion: Private Token Sampling with Homomorphic Encryption. Cryptology ePrint Archive, Paper 2025/2318. https://eprint.iacr.org/2025/2318

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

[23] Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. 2025. THOR: Secure Transformer Inference with Homomorphic Encryption. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security* (Taipei, Taiwan) *(CCS '25)*. Association for Computing Machinery, New York, NY, USA, 3765–3779. doi:10.1145/3719027.3765150

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

[25] Dongjin Park, Eunsang Lee, and Joon-Woo Lee. 2025. Powerformer: Efficient and High-Accuracy Privacy-Preserving Language Model with Homomorphic Encryption. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, Vienna, Austria, 11090–11111. doi:10.18653/v1/2025.acl-long.543

[26] Donghwan Rho, Taeseong Kim, Minje Park, Jung Woo Kim, Hyunsik Chae, Ernest K. Ryu, and Jung Hee Cheon. 2025. Encryption-Friendly LLM Architecture. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=pbre0HKsfE

[27] Donghwan Rho, Sieun Seo, Hyewon Sung, Chohong Min, and Ernest K. Ryu. 2025. Traveling Salesman-Based Token Ordering Improves Stability in Homomorphically Encrypted Language Models. arXiv:2510.12343 [cs.LG] https://arxiv.org/abs/2510.12343

[28] Lorenzo Rovida and Alberto Leporati. 2024. Transformer-based Language Models and Homomorphic Encryption: An Intersection with BERT-tiny. In *Proceedings of the 10th ACM International Workshop on Security and Privacy Analytics* (Porto, Portugal) *(IWSPA '24)*. Association for Computing Machinery, New York, NY, USA, 3–13. doi:10.1145/3643651.3659893

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[30] Linhan Yang, Jingwei Chen, Wangchen Dai, Shuai Wang, Wenyuan Wu, and Yong Feng. 2025. ARION: Attention-Optimized Transformer Inference on Encrypted Data. Cryptology ePrint Archive, Paper 2025/2271. https://eprint.iacr.org/2025/2271

[31] Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. 2024. Secure Transformer Inference Made Non-interactive. Cryptology ePrint Archive, Paper 2024/136. https://eprint.iacr.org/2024/136

[32] Linru Zhang, Xiangning Wang, Jun Jie Sim, Zhicong Huang, Jiahao Zhong, Huaxiong Wang, Pu Duan, and Kwok Yan Lam. 2025. MOAI: Module-Optimizing Architecture for Non-Interactive Secure Transformer Inference. Cryptology ePrint Archive, Paper 2025/991. https://eprint.iacr.org/2025/991