SECURE TRANSFORMER INFERENCE

Mu Yuan, Lan Zhang, Xiang-Yang Li
University of Science and Technology of China
Hefei, China
ym0813@mail.ustc.edu.cn, {zhanglan, xiangyangli}@ustc.edu.cn

ABSTRACT

We present a three-party protocol that can protect both Transformer parameters and user data during the inference phase. For each feedforward inference process, our protocol only introduces permutation computation of input and output data on the user side. Our protocol, Secure Transformer Inference Protocol (STIP), can be applied to real-world services like ChatGPT.

Keywords inference · large language model · permutation · secure protocol · three-party · Transformer

1 Introduction

Applications of Transformer models are exploding, e.g., ChatGPT [1]. Security is critical to Transformer-based services, which determines whether applications can be scaled to privacy-sensitive areas like cloud copilot for proprietary code and documents [2].

Existing work [3, 4] studied this problem under the classic secure multi-party computing framework. Using encryption and decryption methods requires approximation of complex nonlinear layers and introduces heavy computational overhead. In this work, we propose a three-party protocol using permutation to protect both model parameters and user data without any approximation of Transformer models.

2 Formalization

Let $x \in \mathbb{R}^{n\times d}$ denote the input where n is the sequence length (e.g., the number of tokens) and d is the model dimension. We define a Transformer block [5] as a function $f_\theta : \mathbb{R}^{n\times d} \mapsto \mathbb{R}^{n\times d}$ with trainable parameters θ. Then the Transformer inference, i.e., $f_\theta(x) = y$, is computed as follows:

$$Q = xW_q,\ \ K = xW_k,\ \ V = xW_v, \qquad W_q, W_k, W_v \in \mathbb{R}^{d\times d}, \tag{1}$$

$$u = \text{softmax}\left(\frac{QK^T}{\sqrt{k}} + M\right)VW_o, \qquad M \in \mathbb{R}^{n\times n}, W_o \in \mathbb{R}^{d\times d}, \tag{2}$$

$$v = \text{LayerNorm}(u + x; \gamma_1, \beta_1), \qquad \gamma_1, \beta_1 \in \mathbb{R}^d, \tag{3}$$

$$z = \text{ReLU}(vW_1)W_2, \qquad W_1 \in \mathbb{R}^{d\times m}, W_2 \in \mathbb{R}^{m\times d}, \tag{4}$$

$$y = \text{LayerNorm}(z + v; \gamma_2, \beta_2), \qquad \gamma_1, \beta_1 \in \mathbb{R}^d, \tag{5}$$

where k is a constant equal to d divided by the number of attention heads, M denotes the mask which is an all-zero matrix in the encoder and a matrix whose upper right corner (not including the diagonal) is negative infinity in the decoder. The parameter θ consists of attention weights (W_q, W_k, W_v, W_o), feedforward weights (W_1, W_2) and LayerNorm weights (γ, β).

3 Protocol

Let $\pi \in \{0,1\}^{d \times d}$ denote a permutation matrix. We transform the parameters θ as follows:

$$W'_q = \pi^T W_q, \;\; W'_k = \pi^T W_k, \;\; W'_v = \pi^T W_v, \;\; W'_1 = \pi^T W_1$$
$$W'_o = W_o \pi, \;\; W'_2 = W_2 \pi, \;\; \gamma'_1 = \gamma_1 \pi, \;\; \beta'_1 = \beta_1 \pi, \;\; \gamma'_2 = \gamma_2 \pi, \;\; \beta'_2 = \beta_2 \pi.$$

Let θ' denote the transformed parameters, we have:

Theorem 1. $f_{\theta'}(x\pi) = f_\theta(x)\pi$.

Proof. First, we prove that LayerNorm$(x\pi; \gamma\pi, \beta\pi)$ =LayerNorm$(x; \gamma, \beta)\pi$. The LayerNorm function is defined for $x \in \mathbb{R}^{n \times d}$ by

$$\text{LayerNorm}(x; \gamma, \beta) = \gamma \circ \frac{x - \mu_x}{\sigma_x} + \beta, \qquad\qquad \gamma, \beta \in \mathbb{R}^d,$$

where $\circ$ denotes the Hadamard (element-wise) product operator. Since μ_x and σ_x are computed by rows, $\mu_{x\pi} = \mu_x$ and $\sigma_{x\pi} = \sigma_x$. Therefore,

$$\text{LayerNorm}(x\pi; \gamma\pi, \beta\pi) = \gamma\pi \circ \frac{x\pi - \mu_x}{\sigma_x} + \beta\pi = \left(\gamma \circ \frac{x - \mu_x}{\sigma_x} + \beta\right)\pi = \text{LayerNorm}(x; \gamma, \beta)\pi.$$

Then, since $\forall \pi, \pi\pi^T = I$:

$$\begin{aligned}
&Q' = x\pi\pi^T W_q = xW_q = Q,\\
&K' = x\pi\pi^T W_k = xW_k = K,\\
&V' = x\pi\pi^T W_v = xW_v = V,\\
&u' = \text{softmax}\left(\frac{Q'K'^T}{\sqrt{k}} + M\right) V' W_o \pi = \text{softmax}\left(\frac{QK^T}{\sqrt{k}} + M\right) V W_o \pi = u\pi,\\
&v' = \text{LayerNorm}(u' + x\pi; \gamma_1, \beta_1) = \text{LayerNorm}(u\pi + x\pi; \gamma_1, \beta_1) = \text{LayerNorm}((u + x)\pi; \gamma_1, \beta_1) = v\pi,\\
&z' = \text{ReLU}(v'\pi W_1) W_2 \pi = \text{ReLU}(v\pi\pi^T W_1) W_2 \pi = \text{ReLU}(vW_1) W_2 \pi = z\pi,\\
&y' = \text{LayerNorm}(z' + v'; \gamma_2, \beta_2) = \text{LayerNorm}(z\pi + v\pi; \gamma_2, \beta_2) = \text{LayerNorm}((z + v)\pi; \gamma_2, \beta_2) = y\pi,
\end{aligned}$$

i.e., $f'_\theta(x\pi) = y' = y\pi = f_\theta(x)\pi$. □

Leveraging theorem 1, we present a three-party protocol, named Secure Transformer Inference Protocol (STIP):

- Party-1 (P_1): Model developer (e.g., OpenAI) that owns the original Transformer model f_θ.
- Party-2 (P_2): Cloud computing platform (e.g., Azure) that owns the computing hardware.
- Party-3 (P_3): Users that own private input (e.g., prompt token embedding) and output (e.g., response token logits).

Algorithm 1: Secure Transformer Inference Protocol

Initialization phase:
  P_1 randomly generate $\pi \in \mathbb{R}^{d \times d}$;
  P_1 transform f_θ to $f_{\theta'}$ using π;
  P_1 send $f_{\theta'}$ to P_2 and send π to P_3;
Inference phase:
  P_3 transform x to $x' = x\pi$ and send x' to P_2;
  P_2 compute $f_{\theta'}(x') = y'$ and send y' to P_3;
  P_3 de-transform y' by computing $y'\pi^T$ and get $y\pi\pi^T = y$.

Security analysis. Consider P_1 as the attacker against user data x, y, since P_1 cannot get access to $x\pi$ and $y\pi$, P_1 cannot recover x, y although it has π. Consider P_2 as the attacker against model parameters θ and user data x, y, since P_2 has $W\pi$ and $x\pi$, the possibility it guess the correct π is $1/(d!)$. In practice, d is typically larger than 512, e.g., $d = 4096$ in llama [6], so the probability of a successful attack is negligible. Consider P_3 as the attacker against model parameters θ, since P_3 cannot get access to θ', P_3 cannot recover θ although it has π.

4 Discussion

Row-wise permutation. Our protocol permutes x in the column dimension, so a natural question is: What about doing row-wise permutation? In fact, the permutation equivariance property ($f(\pi x) = \pi f(x)$) in the sequence length dimension (row-wise) has been proved for Transformer encoder [7]. For the encoder attention layer:

$$\text{EncAttn}(\pi x) = \text{softmax}\left(\frac{\pi x W_q W_k^T x^T \pi^T}{\sqrt{k}}\right)\pi x W_v W_o = \pi\text{softmax}\left(\frac{x W_q W_k^T x^T}{\sqrt{k}}\right)\pi^T \pi x W_v W_o = \pi\text{Attn}(x).$$

However, due to the mask inside the decoder, attention computation on row-wise permuted data cannot return recoverable output:

$$\text{DecAttn}(\pi x) = \text{softmax}\left(\frac{\pi x W_q W_k^T x^T \pi^T}{\sqrt{k}} + M\right)\pi x W_v W_o \neq \pi\text{DecAttn}(x).$$

A quick fix is to send a transformed $M' = \pi M \pi^T$ to the cloud computing platform party. However, since the value of M is fixed (the upper right corner is negative infinity, and the rest are 0), the cloud computing platform can easily recover the permutation π, which will result in loss of protection.

RMSNorm. Llama [6] uses RMSNorm [8] instead of LayerNorm. Now we prove that $\text{RMSNorm}(x\pi; \gamma\pi) = \text{RMSNorm}(x; \gamma)\pi$. The RMSNorm function is defined for $x \in \mathbb{R}^{n\times d}$ by

$$\text{LayerNorm}(x; \gamma) = \gamma \circ \frac{x}{\sqrt{\frac{1}{n}\sum_i x_i^2}}, \qquad \gamma \in \mathbb{R}^d,$$

where $\circ$ denotes the Hadamard (element-wise) product operator. Since $\sum_i x_i^2$ is computed by rows, $\sum_i (x\pi)_i^2 = \sum_i x_i^2$. Therefore,

$$\text{RMSNorm}(x\pi; \gamma\pi) = \gamma\pi \circ \frac{x\pi}{\sqrt{\frac{1}{n}\sum_i (x\pi)_i^2}} = \left(\gamma \circ \frac{x}{\sqrt{\frac{1}{n}\sum_i x_i^2}}\right)\pi = \text{RMSNorm}(x; \gamma)\pi.$$

SwiGLU feedforward. Llama [6] uses SwiGLU [9] instead of ReLU in feedforward layers. Let $\text{FFN}_{\text{SwiGLU}}$ denote the feedforward layers using SwiGLU, which is defined by:

$$\text{FFN}_{\text{SwiGLU}}(x) = ((xW_1)\text{sigmoid}(xW_1)xW_3)W_2, \qquad W_1, W_3 \in \mathbb{R}^{d\times m}, W_2 \in \mathbb{R}^{m\times d}.$$

We transform parameters as follows:

$$W_1' = \pi^T W_1, \;\; W_3' = \pi^T W_3, \;\; W_2' = W_2\pi,$$

and let $\text{FFN}'_{\text{SwiGLU}}$ denote the transformed function. Now we prove that $\text{FFN}'_{\text{SwiGLU}}(x\pi) = \text{FFN}_{\text{SwiGLU}}(x)\pi$:

$$\begin{aligned}\text{FFN}'_{\text{SwiGLU}}(x\pi) &= ((x\pi\pi^T W_1)\text{sigmoid}(x\pi\pi^T W_1)x\pi\pi^T W_3)W_2\pi \\ &= ((xW_1)\text{sigmoid}(xW_1)xW_3)W_2\pi = \text{FFN}_{\text{SwiGLU}}(x)\pi.\end{aligned}$$

Applicable scope. In fact, STIP is applicable to models that are built with any global matrix multiplication-based (e.g., attention and feedforward) layers and row-wise (e.g., LayerNorm) layers. To give some counterexamples, STIP cannot be applied to convolutional layers.

Our test code of STIP for the original Transformer [5] and llama [6] can be found in `https://github.com/yuanmu97/secure-transformer-inference`.

5 Conclusion

In this paper, we present a secure protocol (STIP) for serving Transformer models in a three-party setting.

Acknowledgments

We would like to thank Yihang Cheng, Miao-Hui Song, Ning-Kang Zhang, Puhan Luo, and Junyang Zhang for their contributions to the protocol design.

References

[1] OpenAI. Chatgpt. https://openai.com/blog/chatgpt, 2022.

[2] Microsoft. Microsoft 365 copilot. https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/, 2023.

[3] 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.

[4] 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.

[5] 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.

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

[7] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

[8] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

[9] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.