Proof-of-Concept for Private Local-to-Cloud LLM Chat via Trusted Execution Environments

Avanika Narayan [*1] Dan Biderman [*1 2] Christopher Ré [1]

Abstract

Cloud-based LLM assistants pass every prompt through cloud servers in plaintext leaving personal information open to inspection by cloud providers and any malicious actors with access to their servers. Current privacy techniques either degrade quality or are several orders of magnitude slower. In contrast, Trusted Execution Environments (TEEs) offer a practical path forward, taking a hardware-based approach. We explore recent TEE-based virtual machines with confidential NVIDIA H100 and AMD SEV-SNP CPUs. Naive Pytorch use inside this TEE incurs a 1.87× slowdown due to CPU-GPU encryptions. Moreover, there is a lack of open-source communication protocols between a local client and such a remote TEE. In response, we propose TEECHAT, a research prototype that (1) binds a local client to a remote TEE hosting an LLM, via attestation and key exchange, (2) secures communication with full end-to-end encryption, and (3) minimizes overhead with targeted kernel and I/O optimizations. For models over 30B parameters, TEECHAT adds just 1% latency—showing that LLM inference inside TEEs is already practical[1].

1. Introduction

Large Language Models (LLMs) now serve hundreds of millions of requests each day, many containing medical, legal, or proprietary information (Singh, 2025; Mireshghallah et al., 2024). Today, these prompts and responses are usually processed *in plaintext* on cloud machines outside the user's control, exposing them to cloud providers, infrastructure operators and potential adversaries with access to the machine.

Existing defences fall at two extremes. One is removal of personally identifiable information (PII) before transmission to the remote LLM server. This approach enables standard LLM computation, however, it may leave residues and often degrades quality (Siyan et al., 2024). The second is Fully Homomorphic Encryption (FHE), which computes directly on encrypted data, yet its $100\times$–$10{,}000\times$ slowdown remains impractical for real-time chat (Gilad-Bachrach et al., 2016; Riazi et al., 2019; Lloret-Talavera et al., 2021).

Trusted Execution Environments (TEEs) provide a middle ground. TEEs create hardware-isolated regions ("enclaves") where data and code are hidden from the host system. NVIDIA's H100 "Hopper" GPU is the first to extend this model to the entire accelerator – covering both memory and compute – forming a so-called confidential GPU (CGPU). The CGPU is nested within a CPU-based TEE, which decrypts the sensitive text, runs the LLM server that launches GPU kernels, and encrypts all CPU–GPU communication, adding a second layer of protection. Contemporary confidential VMs are capable of serving LLMs with tens of billions of parameters while preserving standard TEE guarantees (see Section 2.3).

We ask: *to what extent can TEEs enable accurate, end-to-end encrypted communication between a local client and an LLM in a cloud-hosted CGPU, without incurring prohibitive latency?*

Our first attempt to connect a local client to a CGPU-hosted LLM, revealed two core challenges:

- **Without model-specific kernel optimization, confidential VMs are slower than standard VMs.** A confidential machine with nested CPU-GPU enclaves is $1.87\times$ slower than a standard VM with GPU when running PyTorch code (Paszke et al., 2019; Wolf et al., 2020). The extra CPU-GPU encryptions start to take a toll on performance when launching multiple kernels (up to 67% slower) or performing frequent CPU-GPU I/O (up to 1,200% slower).

[*]Equal contribution [1]Department of Computer Science, Stanford University [2]Department of Statistics, Stanford University. Correspondence to: Avanika Narayan <avanikan@stanford.edu>, Dan Biderman <biderman@stanford.edu>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

[1]This work was first published as a blog post (Biderman et al., 2024), which was subsequently cited in a white paper by Anthropic and Pattern Labs (Anthropic, 2024).

- **Local–cloud gap.** Existing TEEs secure only the remote LLM side, leaving the communication with the local client unprotected. Currently, no open-source solution secures the entire client-to-cloud interaction.

Motivated by these limitations, we propose TEECHAT, a *prototype* for a secure communication protocol between a local client and an LLM running in cloud confidential VM. TEECHAT further extends the core guarantees of TEEs to the local-cloud setup while adding *negligible* overhead.

We benchmark the overall performance of TEECHAT, finding that after minimizing kernel launches and I/O (using standard, optimized LLM inference packages), the protocol latency is dominated by LLM inference latency; encryption and authentication add just 2ms. We find that for models larger than 30B parameters with batch sizes under 32, the overhead in latency and throughput is under 1% compared to a standard chat.

To summarize, this work makes the following contributions:

- We study the performance of LLMs inside confidential GPU-powered VMs, identifying CPU-GPU communications as the major bottleneck.
- We propose an end-to-end prototype for key-exchange, remote attestation, and encrypted streaming with minimal latency cost and no quality degradation.

This study is a proof-of-concept, not a secure system for deployment, seeking to understand the security-performance tradeoffs in emerging confidential VMs.

2. Preliminaries

We describe the local-cloud chat setup (see §2.1), the threat model (see §2.2) and secure hardware details (see §2.3).

2.1. Setup: Local–Cloud Chat

We model a single interactive session between a *user* U and a large language model (LLM) $\mathcal{M}$ hosted in the cloud by provider $\mathcal{P}$. We detail the system components as follows:

1. **User device** D. A trusted local endpoint (e.g., laptop or phone) controlled by U. It sends prompts, receives answers, and stores all sensitive information decrypted.
2. **Cloud system** $\mathsf{S} = \langle \mathsf{R}, \mathsf{G}, \mathcal{M} \rangle$. Hosted by the provider $\mathcal{P}$, and serving an **LLM** $\mathcal{M}$.
   (a) **CPU.** R. Runs the LLM-server process, holding the model code. The CPU handles request scheduling, tokenization, and launches GPU kernels.
   (b) **GPU Accelerator** G. Executes all inference kernels, and holds $\mathcal{M}$'s parameters and intermediate activations.
3. **Communication channel** $\mathcal{C}$. A bidirectional network link between D and $\mathcal{M}$.

2.2. Threat Model

In line with prior literature on enclave security (Costan & Devadas, 2016; Lee et al., 2020; Volos et al., 2018), we assume an adversary either with access to the VM or to the communication channel with it.

Adversary Model. We do not trust the cloud provider and any actors running processes on the system. Moreover, we do not trust the datacenter operators, who may be third-party vendors with physical access to the hardware. Our goal is to keep the LLM's inputs, intermediate activations, and outputs confidential from these parties.

Trusted Components. First, we assume the user device is secure throughout. Second, we trust the silicon manufacturers and their firmware, i.e., NVIDIA's H100 confidential-mode stack, and AMD SEV-SNP, as we need hardware-guarantees to perform TEE attestation, following previous work (Costan & Devadas, 2016; Lee et al., 2020).

2.3. Trusted Execution Environments & Confidential GPUs

Trusted Execution Environment (TEE) A TEE is a hardware-enforced enclave that keeps its code and data private and tamper-free even if the host OS or hypervisor is compromised. TEEs offer three guarantees: **confidentiality**—code and data inside the enclave are invisible to any external process, including privileged software (Costan et al., 2016); **integrity**—no unauthorized entity can alter the enclave's computation or memory state (Sabt et al., 2015); and **authenticity**—via *remote attestation*, the enclave can prove to a verifier that approved code is running on genuine hardware (McKeen et al., 2013). Together, these guarantees establish a trusted boundary that holds even in hostile cloud environments (Costan et al., 2016).

Confidential GPUs The NVIDIA Hopper–generation H100 is the first commercially available GPU with native *confidential-computing* support, effectively turning the accelerator into a *GPU-scale TEE* (NVIDIA, 2023). A dedicated on-die security processor establishes a hardware root of trust, verifies the GPU firmware and loaded kernels (secure boot), and transparently encrypts every transaction between the streaming multiprocessors (SMs), on-package caches, and high-bandwidth memory (HBM) (memory encryption). Once the remote-attestation handshake completes, the security processor derives keys that only the session owner can access; even a privileged cloud operator snooping PCIe links or probing HBM sees only ciphertext (NVIDIA, 2023; 2024). Hopper's Secure Execution Environment (SEE) mirrors the classic TEE guarantees (NVIDIA, 2022).

Confidential virtual machines CGPUs are often hosted inside confidential virtual machines, such as Azure ND H100 v5, creating two concentric TEEs. The process code is run via a CPU-based enclave (SEV-SNP) that encrypts all guest-CPU code and DRAM against the host OS. The CPU code may launch kernels on the CGPU, and any information transferred between them CPU and GPU is encrypted.

3. Related Works

A growing line of work executes Transformers on *fully homomorphically encrypted* (FHE) inputs, e.g. THOR (Moon et al., 2024) and NEXUS (Zhang et al., 2024), achieving strong cryptographic privacy but at the cost of minute–scale latencies (10–600× slower than plaintext). In contrast to this line of research, our work attempts to secure LLM inference by leveraging hardware-based security features. Additionally, there exists work that studies the efficacy of running transformer based architectures inside secure CPU enclaves. These works study transformers inside secure enclaves and partitioning layers to balance speed and security (Chang & Chen, 2025; Lee et al., 2025) for both inference and training. In our work, we solely focus on REMOTE CPU-GPU enclaves and are concerned with facilitating confidentiality across local to cloud chat environments. Another line of work handles the privacy challenge by removing sensitive tokens before passing to an insecure cloud LLM (Zeng et al., 2024). In contrast, this work keeps the *entire* prompt encrypted, and utilizes hardware security to ensure that no plaintext ever leaves the secure enclave. Finally, there exists a line of work around industrial secure cloud deployments (Inc., 2024) that utilizes custom hardware to ensure cloud privacy (Inc., 2024). However, this work is not open and it is thus unclear how compatible it is with NVIDIA H100 CGPUs.

4. TEECHAT: An End-to-end Encrypted Chat Protocol

Our goal is to satisfy the three guarantees of TEEs across a communication between a user device D and a cloud system S, which is a confidential virtual machine serving an LLM $\mathcal{M}$. TEECHAT has four main steps as seen in Figure 1:

1. **Ephemeral Key Exchange**. The client device (D) generates a fresh key pair for each chat session and runs a Diffie–Hellman key exchange with the *CPU enclave*, deriving a session-unique shared secret *over an HTTPS channel* (Step 1 in Fig. 1).
2. **Remote Attestation**. Before user data is released, D requests a hardware-signed quote that covers *both* the AMD SEV-SNP CPU enclave and the NVIDIA H100 CGPU enclave. The quote proves that (i) the devices are genuine, (ii) they run in confidential-computing mode, and (iii) the VM launch measurement matches a reference hash we publish (Step 2).
3. **Encrypted & Authenticated Transport**. All subsequent traffic on the channel $\mathcal{C}$ is *end-to-end encrypted and signed* under the session key. We use monotonically increasing nonces for replay protection; the current prototype re-uses the HTTPS framing for convenience (Step 3).
4. **Nested-TEE Inference**. Ciphertexts are decrypted only inside the CPU enclave, which then launches kernels on the confidential H100 GPU. Plaintext prompts, activations, and responses never leave this nested TEE; outputs are immediately re-encrypted and signed before exiting the enclave (Step 4).

In **??**, we provide further details on the protocol and how it provides the TEE guarantees.

5. TEECHAT Protocol Performance

We begin by measuring the performance overheads of GPU serving within a TEE (see Section 5.1). We then evaluate the performance of the end-to-end protocol, specifically measuring (1) where the performance bottlenecks exist and (2) what overhead is introduced by running in a confidential vs. standard GPU-powered virtual machine (see Section 5.2).

Experimental Set-up: We perform all our analysis using an Azure ND H100 v5 confidential VM as the cloud system (S), containing an *NVIDIA-H100 Confidential GPU* (CGPU) as our accelerater (G) and AMD SEV-SNP CPU. We use *Apple M1-Max* laptop as the user device (D). Because Azure's virtualization stack is closed-source, our measurements inherit trust in Azure's launch stack and operating system. In principle, if users can inspect the virtualization stack and operating system, they do not have to trust a cloud provider. For our benchmarking we use the LLAMA3.2-3B-INSTRUCT (Grattafiori et al., 2024), QWEN3-8B (Yang et al., 2025) and QWEN3-32B (Yang et al., 2025). We summarize the takeaways of our analysis as follows:

- **Dominant cost.** End-to-end overhead is dominated by *remote* model inference; message-layer encryption and authentication add only $\approx$10 ms.
- **Overhead vs. model size.** Remote-side overhead is ˜17% for $\leq$3B models, ˜8% for 3–10B, and 1–2% for $>$10B.

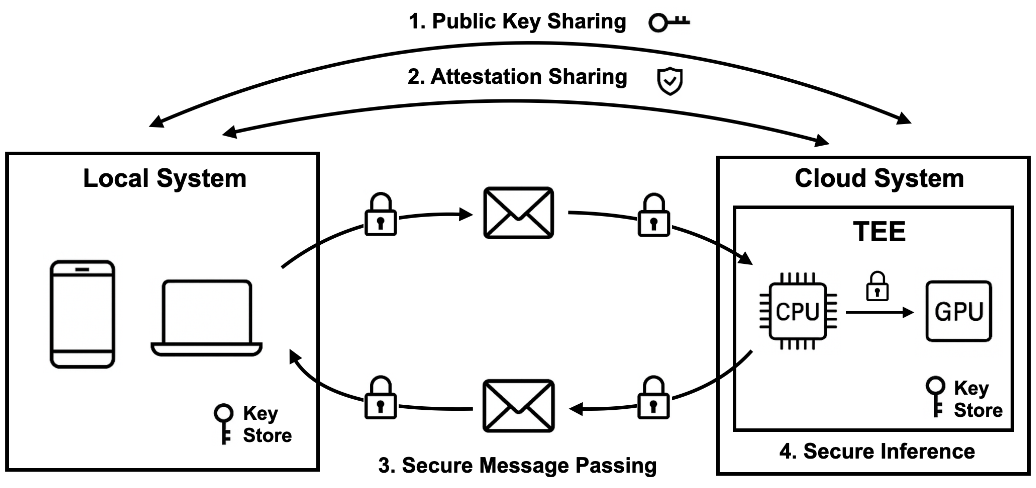

Figure 1. **TEECHAT protocol**: a secure end-to-end encrypted chat with 4 key steps—(1) ephemeral key exchange generates a unique session secret, (2) remote attestation verifies hardware authenticity, (3) encrypted message passing ensures confidentiality and authenticity, and (4) enclave-bound inference securely processes and re-encrypts messages.

- **Batching.** The remote LM is optimally utilized at a batch size of < 16 (see §5.1).
- **TEE bottlenecks.** Kernel launches and CPU – GPU transfers are the primary bottlenecks (see §5.1); fused kernels and reduced round-trips yield optimal speed.

5.1. CGPU Benchmarking

Benchmark	Standard (ms)	CGPU (ms)	Overhead
GEMM 4096×4096 FP16	0.06	0.10	+67%
Host↔GPU 512 MB	1.85	24.58	+1,230%

Table 1. **Operation overhead analysis**: Latency overhead in a confidential vs. standard virtual machine for two operations: matrix multiplies and CPU-GPU data transfers.

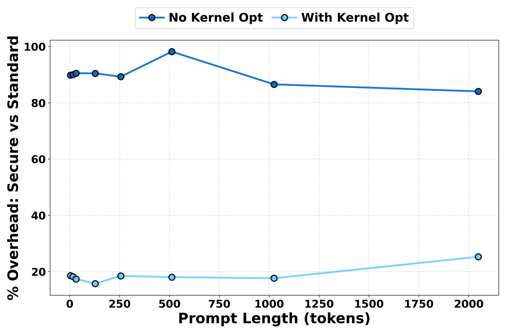

Figure 2. **Effects of kernel optimization**. Overhead falls by $\sim 5.6\times$ when three optimizations are enabled: (i) FlashAttention-2 which keeps Q, K, V in SRAM; (ii) CUDA-Graph replay captures the first decode step and re-uses it every token; (iii) On-GPU fused sampling executes softmax $\rightarrow$ top-p/k entirely on device.

We benchmark a single NVIDIA H100 within and without a TEE (within a TEE, it uses AMD SEV-SNP). To pinpoint latency sources we run two micro-benchmarks that stress complementary subsystems: (i) *compute-bound* GEMM (4096×4096, FP16) and (ii) *I/O-bound* 512 MB host↔GPU round-trip copy. Each benchmark runs 100 iterations; we report median latency. Table 1 summarizes the latency overhead for compute and I/O-bound workloads. We observe that kernel operations incur a launch penalty and data transfer across CPU incurs heavy slow-downs. In Figure 2 we benchmark three kernel-level optimizations for LLAMA-3.2-3B while generating 128 tokens for various pre-fill lengths. Replacing the standard SDPA kernel with FlashAttention-2 reduces memory usage from quadratic to linear in sequence length (Dao et al., 2023). Wrapping the first decode call in a CUDA Graph lets the remaining 127 steps run as graph replays, eliminating all CPU launch overhead. Finally, executing softmax and top-k sampling entirely on-device with FlashInfer removes host-device transfers and further trims latency (Wu et al., 2024).

Optimal Batch Sizes We observe that batch sizes < 16 are optimal (see Figure 3); beyond this point, CGPU kernel-launch overhead and memory-bandwidth contention outweigh batching benefits. Slowdown decreases with larger models: 28.5% for 3B parameters versus 4.6% for 8B parameters.

5.2. End-to-End Protocol Performance

Next, we measure the end-to-end protocol performance, focusing on each of the 4 steps of TEECHAT (see Figure 1).

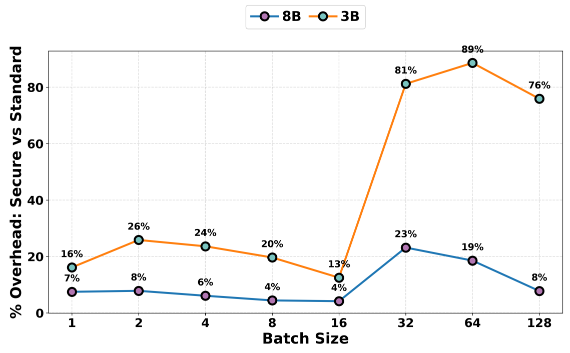

Figure 3. **Overhead as a function of batch size.** Batching leads to overheads in CGPUs. Overheads are less for larger models. Batch size of 16 is optimal.

t

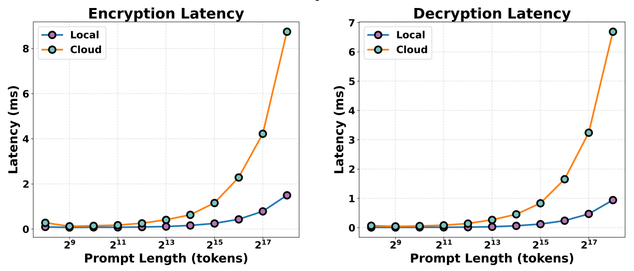

Figure 4. **Message encryption overheads**. Micro-benchmark of encryption/decryption and verification overhead on the local host (M1–Max) and remote CGPU node. (Left) decryption speeds and (Right) encryption speeds across local and remote hardware. Latency is negligible even for very long contexts ($< 2^{15}$ tokens)

5.2.1. Key Exchange and Remote Attestation

We measure the total time taken for key exchange and remote attestation. We find that a full key exchange plus GPU attestation runs once per session and adds only <0.2 ms of latency (note that in this analysis we did not time the CPU attestation).

5.2.2. Encrypted Message Passing

We find that encrypted communication adds negligible latency.

We measure the time (ms) required for encryption and signing ("encrypt") and for decryption and verification ("decrypt"). For a 65.5k-token payload, local encryption consumes 0.55 ± 0.04 ms and decryption 0.36 ± 0.07 ms. On the remote host, encryption requires 2.28 ± 0.12 ms and decryption 1.65 ± 0.05 ms (See Figure 4 for costs incurred at different prefill lengths). We find that encryption involves a 2.2X overhead and decryption 1.6X overhead, when averaging across inputs of length 512 to 8192 tokens.

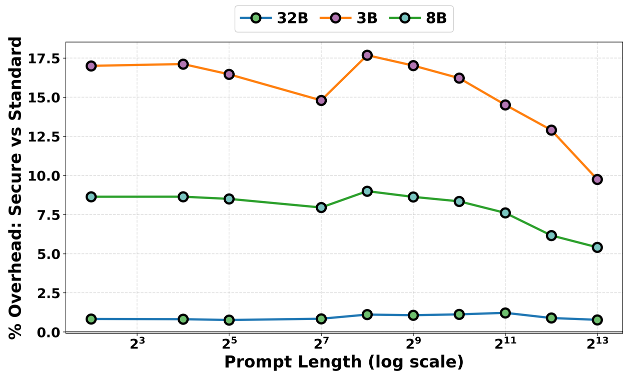

Figure 5. **Inference overhead as a function of prefill length**. Overhead is negligible for large models $> 30B$ and decreases as prefill length increases (as time spent on LLM inference grows faster than the fixed security costs).

5.2.3. Enclave-Bound Inference

We find that TEE-based inference overheads become negligible for models larger than 30B parameters.

We measure both the latency and queries per second (QPS) of various workloads. We additionally benchmark efficiency against a standard GPU. We measure latency (prefill sweep, 128-token generation) and QPS (batch sweep, fixed prefill 1024, 48-token generation) (see Figure 5). Averaged across the prefill sweep, TEE adds 77.24 ms (3B) and 31.32 ms (32B) latency—13.9% and 0.94% overhead—respectively. This suggests that overhead reduces as model size increases as compute becomes the predominant cost factor. Similarly, throughput overhead declines from 22% (3B), to 0.03% (32B) relative to a standard inference (see Figure 6).

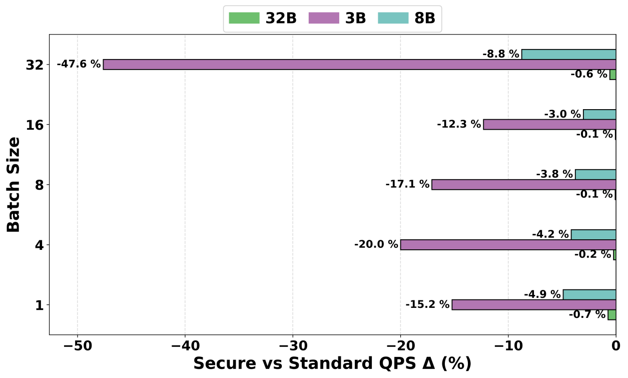

Figure 6. **Query Per Second (QPS) overhead.** Across all model sizes, overhead is lowest for batch size = 16. Overhead is negligible for models $> 30B$ parameters.

6. Conclusion

We present TEECHAT, a research prototype which securely connects local clients to remote TEE-based virtual machines, providing cryptographic guarantees and negligible latency overhead ($<$ 1% for models $>$30B parameters). As models and workloads evolve, future research must continue expanding these security methods to distributed, heterogeneous environments.

References

Anthropic. Confidential inference on trusted vms. Blog Post and White Paper, 2024. URL https://www.anthropic.com/research/confidential-inference-trusted-vms.

Biderman, D., Narayan, A., and Ré, C. Securing private llm inference with trusted hardware. Blog Post, Hazy Research, 2024. URL https://hazyresearch.stanford.edu/blog/2025-05-12-security.

Chang, W. and Chen, M. Secure and efficient transformer inference with TEE. Technical Report, National Taiwan University, March 2025. URL https://arbor.ee.ntu.edu.tw/static/gm/20250303_%E5%BC%B5%E7%B4%8B%E6%85%88.pdf.

Costan, V. and Devadas, S. Intel SGX Explained. Cryptology ePrint Archive, Paper 2016/086, 2016. URL https://eprint.iacr.org/2016/086.

Costan, V., Lebedev, I., and Devadas, S. Sanctum: Minimal hardware extensions for strong software isolation. In *25th USENIX Security Symposium (USENIX Security '16)*, pp. 857–874, 2016. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/costan.

Dao, T., Shi, H., Fu, L., Fu, A., Zhao, K., Li, R., Yu, S., Ermon, S., Smith, J., and Stoica, I. Flashattention-2: Faster attention with better parallelism and work partitioning. In *Advances in Neural Information Processing Systems*, 2023.

Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on Machine Learning*, ICML '16, pp. 201–210. PMLR, 2016. URL https://proceedings.mlr.press/v48/gilad-bachrach16.pdf.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Inc., A. Private cloud compute: A new frontier for AI privacy in the cloud. Apple Security Engineering and Architecture Blog, June 2024. URL https://security.apple.com/blog/private-cloud-compute/.

Lee, D., Kohlbrenner, D., Shinde, S., Asanović, K., and Song, D. Keystone: An open framework for architecting trusted execution environments. In *Proceedings of the 15th European Conference on Computer Systems (EuroSys '20)*, 2020.

Lee, J., Wang, Y., Rajat, R., and Annavaram, M. Characterization of gpu TEE overheads in distributed data-parallel ML training. arXiv:2501.11771, 2025. URL https://arxiv.org/abs/2501.11771.

Lloret-Talavera, G., Jorda, M., Servat, H., Boemer, F., Chauhan, C., Tomishima, S., Shah, N. N., and Pena, A. J. Enabling homomorphically encrypted inference for large dnn models. *IEEE Transactions on Computers*, 71(5): 1145–1155, 2021.

McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C. V., Shafi, H., Shanbhogue, V., and Savagaonkar, U. R. Innovative instructions and software model for isolated execution. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP '13)*. ACM, 2013. doi: 10.1145/2487726.2488368.

Mireshghallah, N., Antoniak, M., More, Y., Choi, Y., and Farnadi, G. Trust no bot: Discovering personal disclosures in human-llm conversations in the wild. *arXiv preprint arXiv:2407.11438*, 2024.

Moon, J., Yoo, D., Jiang, X., and Kim, M. THOR: Secure transformer inference with homomorphic encryption. Cryptology ePrint Archive, Paper 2024/1881, 2024. URL https://eprint.iacr.org/2024/1881.

NVIDIA. Nvidia h100 tensor core gpu architecture. https://www.advancedclustering.com/wp-content/uploads/2022/03/gtc22-whitepaper-hopper.pdf, 2022. Section "Security Enhancements and Confidential Computing".

NVIDIA. Confidential computing on nvidia gpus. Technical report, NVIDIA Technical Brief, 2023. URL https://developer.nvidia.com/confidential-computing.

NVIDIA. Hopper architecture: Secure enclave and attestation for h100 gpus. White Paper, 2024. URL https://resources.

nvidia.com/en-us-gpu-architecture/hopper-security.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. URL https://arxiv.org/abs/1912.01703.

Riazi, M. S., Samragh, M., Mishra, P., Koushanfar, F., and Javadi, H. Xonn: Xnor-based oblivious deep neural network inference. *IACR Cryptol. ePrint Arch.*, 2019:171, 2019. URL https://eprint.iacr.org/2019/171.

Sabt, I., Achemlal, M., and Bouallegue, T. Trusted execution environments: A survey on hardware and software security. *IEEE Communications Surveys & Tutorials*, 17(3): 1250–1270, 2015. doi: 10.1109/COMST.2015.2444095.

Singh, S. Chatgpt statistics 2025 – dau & mau data (worldwide). https://www.demandsage.com/chatgpt-statistics/, 2025. Accessed: 2025-05-20.

Siyan, L., Raghuram, V. C., Khattab, O., Hirschberg, J., and Yu, Z. Papillon: Privacy preservation from internet-based and local language model ensembles. *arXiv preprint arXiv:2410.17127*, 2024.

Volos, S., Vaswani, K., and Bruno, R. Graviton: Trusted execution environments on gpus. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18)*, pp. 681–696, 2018.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2020. URL https://arxiv.org/abs/1910.03771.

Wu, Y., Chen, S., Michel, P., Cai, D., and Liu, Y. Flashinfer: Efficient parallel transformer inference kernels. arXiv preprint arXiv:2402.13338, 2024.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report. https://arxiv.org/abs/2505.09388, 2025.

Zeng, Z., Wang, J., Yang, J., Lu, Z., Zhuang, H., and Chen, C. PrivacyRestore: Privacy-preserving inference in large language models via privacy removal and restoration. *arXiv preprint arXiv:2406.01394*, 2024. URL https://arxiv.org/abs/2406.01394.

Zhang, J., Yang, X., He, L., Chen, K., jie Lu, W., Wang, Y., Hou, X., Liu, J., Ren, K., and Yang, X. Secure transformer inference made non-interactive. Cryptology ePrint Archive, Paper 2024/136, 2024. URL https://eprint.iacr.org/2024/136. To appear in NDSS 2025.