Privacy-Preserving LLM Interaction with Socratic Chain-of-Thought Reasoning and Homomorphically Encrypted Vector Databases

Anonymous Author(s)
Affiliation
Address
email

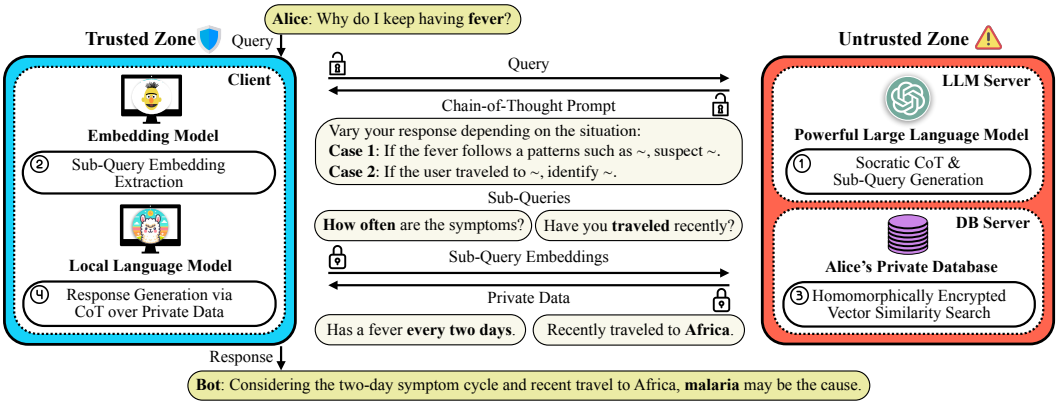

Figure 1: **Overview of our hybrid framework.** Upon receiving a query, a remote LLM generates a Chain-of-Thought (CoT) prompt and sub-queries (Stage 1) which are embedded locally (Stage 2), and used for our encrypted vector search on a remote database (Stage 3). Retrieved records are decrypted and provided with the CoT prompt as context to a local model to generate the final response (Stage 4).

Abstract

Large language models (LLMs) are increasingly used as personal agents, accessing sensitive user data such as calendars, emails, and medical records. Users currently face a trade-off: They can send private records—many of which are stored in remote databases—to powerful but untrusted LLM providers, increasing their exposure risk. Alternatively, they can run less powerful models locally on trusted devices. We bridge this gap: Our **Socratic Chain-of-Thought Reasoning** first sends a generic, non-private user query to a powerful, untrusted LLM, which generates a Chain-of-Thought (CoT) prompt and detailed sub-queries without accessing user data. Next, we embed these sub-queries and perform encrypted sub-second semantic search using our **Homomorphically Encrypted Vector Database** across one million entries of a single user's private data. This represents a realistic scale of personal documents, emails, and records accumulated over years of digital activity. Finally, we feed the CoT prompt and the decrypted records to a local language model and generate the final response. On the LoCoMo long-context QA benchmark, our **hybrid framework**—combining GPT-4o with a local Llama-3.2-1B model—outperforms using GPT-4o alone by up to 7.1 percentage points. This demonstrates a first step toward systems where tasks are decomposed and split between untrusted strong LLMs and weak local ones, preserving user privacy.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

1 Introduction

Large language models (LLMs) are becoming the default backend for personal agents that manage emails, schedule meetings, and process health data [36, 29, 41]. These agents must integrate data from heterogeneous sources using retrieval-augmented generation (RAG) [25]. While forwarding user queries with retrieved data to powerful LLMs enhances performance, it introduces substantial privacy risks [50, 21]. Conversely, restricting operations to local devices significantly degrades performance [30].

Problem: Users face a fundamental trade-off between privacy and utility. Powerful cloud LLMs offer superior reasoning but require exposing private data to untrusted providers. Local models preserve privacy but lack computational capacity for complex reasoning tasks.

We propose a four-stage hybrid framework that partitions tasks between untrusted powerful LLMs and trusted lightweight local models (Figure 1). Our key insight is that many complex queries can be decomposed into: (1) abstract reasoning that doesn't require private data, and (2) contextual retrieval and response generation that can be handled locally.

Socratic Chain-of-Thought Reasoning enables challenging yet non-private queries to be offloaded to powerful external LLMs. When a user asks "Why do I keep having fever?", we send only this generic query to GPT-4o, which generates targeted sub-queries (e.g., "How often are symptoms?" "Recent travel?") and reasoning prompts without accessing private data. **Homomorphically Encrypted Vector Database** enables secure semantic search over encrypted records—the cloud provider executes searches without learning data content.

Our framework operates in four stages: (1) Send generic user query to powerful LLM for chain-of-thought and sub-query generation, (2) Locally embed sub-queries for encrypted search, (3) Execute secure similarity search over encrypted million-scale database in <1 second, (4) Local model generates final response using CoT prompt and decrypted records.

Results: On LoCoMo long-context QA, our hybrid approach with Llama-3.2-1B achieves F1=87.7, surpassing GPT-4o alone (80.6) by 7.1 percentage points and local-only baseline by 23.1 points. This counterintuitive improvement demonstrates the power of structured task decomposition. Our encrypted database achieves >99% accuracy with 5.8× storage overhead and sub-second latency on million-scale collections.

Contributions: (1) First framework enabling privacy-preserving LLM interaction through task decomposition between untrusted and trusted models, (2) Novel Socratic Chain-of-Thought method that improves performance while preserving privacy, (3) Efficient homomorphically encrypted vector database with practical performance, (4) Demonstration that hybrid approaches can outperform monolithic powerful models.

2 Background and Problem Formulation

Large language models (LLMs) increasingly serve as personal assistants, processing sensitive user data such as calendars, emails, and medical records [49, 36]. Effective LLM-based personal assistants require two fundamental capabilities:

(1) Contextual Reasoning: The model must establish clear criteria to accurately interpret user queries in context. For instance, recognizing *a cyclic fever pattern recurring every two days* in combination with *recent travel to Africa* strongly suggests *malaria*. Augmenting such contextual understanding into the reasoning process ensures precise and meaningful conclusions.

(2) Contextual Data Retrieval: The model must determine which contextual data is necessary for comprehensive understanding. As illustrated in Figure 1, a user's query such as *"Why do I keep having fever?"* might not provide enough context to retrieve all necessary records. The model must generate targeted sub-queries to collect comprehensive information, such as travel history that might reveal malaria risk factors [25].

Privacy Problem Formulation: While powerful cloud-based LLMs offer superior reasoning capabilities, they require users to expose private data to untrusted providers [33]. Conversely, local models that preserve privacy lack the computational capacity for complex reasoning tasks. We consider a user with a non-private query whose answer depends on private records stored remotely (As shown in

Figure 1). The local device has limited computational resources insufficient for complex reasoning, while powerful cloud LLMs cannot be trusted with sensitive data [45].

Threat Model: We protect against three adversaries: (1) the LLM provider who receives user queries, (2) the database provider storing encrypted records [5], and (3) external attackers who may compromise these services [19]. Even with standard encryption, providers typically hold decryption keys, enabling potential privacy breaches through insider threats or security compromises [7, 17].

Privacy Goal: User data must remain encrypted outside the trusted local environment, with decryption keys never leaving the user's control. The system must enable complex reasoning and efficient retrieval while ensuring that untrusted components cannot access plaintext private data [14, 38].

3 Privacy-Preserving Framework with Socratic Chain-of-Thought Reasoning

Our framework separates computation into trusted and untrusted zones to balance privacy and performance (Figure 1). The trusted zone (left) hosts a lightweight LLM and embedding model with exclusive access to decryption keys. The untrusted zone (right) comprises cloud providers hosting: (1) a powerful LLM for abstract reasoning, and (2) an encrypted vector database using homomorphic encryption [14].

3.1 Framework Operation

Consider the medical example in Figure 1: when a user asks "Why do I keep having fever?", our framework operates as follows:

Stage 1 - Socratic Reasoning: The generic query is sent to GPT-4o, which generates:

- **Chain-of-Thought prompt:** "Vary response by situation: Case 1: If fever follows patterns, suspect recurring illness. Case 2: If user traveled recently, identify location-specific diseases."
- **Sub-queries:** "How often are symptoms?" and "Recent travel history?"

Stage 2 - Local Embedding: Sub-queries are embedded locally and prepared for encrypted search without exposing content to cloud providers.

Stage 3 - Encrypted Search: Our homomorphically encrypted database executes similarity search over encrypted user records, retrieving top-k matches like "Has fever every two days" and "Recently traveled to Africa" while maintaining encryption.

Stage 4 - Local Response: The local Llama model combines the CoT prompt and decrypted records to generate: "Considering the two-day cycle and Africa travel, malaria may be the cause."

This decomposition ensures powerful models operate only on non-private data while private records remain encrypted outside the trusted zone. The approach provides both active control (users manage what reaches remote models) and passive control (cryptographic protection ensures data security even with user errors).

3.2 Key Properties

Privacy Guarantees: Private data never leaves the trusted zone in plaintext. Even if users accidentally send sensitive queries, the database remains encrypted with keys held exclusively locally.

Performance Benefits: Delegating complex reasoning to powerful models while keeping private retrieval local often improves performance through structured test-time computation compared to monolithic approaches.

4 Homomorphically Encrypted Vector Database

Personal AI assistants require large-scale user data for effective retrieval, but cloud storage introduces privacy risks. Our homomorphically encrypted vector database enables semantic search over private data without exposing plaintext to untrusted servers.

Challenge: Standard homomorphic encryption approaches suffer from high computational overhead and cannot support dynamic updates efficiently. Existing methods like CHAM [37] require expensive preprocessing that becomes impractical when users frequently add personal data.

Our Approach: We develop a novel inner product protocol that separates query and key operations, enabling efficient caching of encrypted vectors while supporting constant-time insertions and deletions. Key innovations include: (1) Query-key decoupling that allows precomputation independent of queries, (2) Butterfly decomposition reducing automorphism complexity, (3) SIMD-style operations in encrypted domain, and (4) Seed-based ciphertext generation for compact storage.

Security: Our system provides 128-bit IND-CPA security via CKKS encryption [10] for vectors and AES-256 for data values, with quantum-resistant guarantees [5]. The database provider cannot access plaintext data or learn query patterns.

Performance: Our encrypted database achieves sub-second semantic search across one million 768-dimensional vectors with >99% recall accuracy compared to plaintext search. Storage overhead is 5.8× with linear scalability. The system outperforms prior work (CHAM) by 37× on million-scale searches through optimized key-switching that scales with vector length rather than matrix size.

Detailed algorithms, security analysis, and technical optimizations are provided in Appendix A.

5 Experiments

We evaluate our framework on LoCoMo [31] (personal assistant scenarios) and MediQ [27] (medical consultations), comparing against local-only baselines (Llama-3.2-1B/3B) and remote-only baselines (GPT-4o, Gemini-1.5-Pro, Claude-3.5-Sonnet). All experiments use DRAGON [28] for embeddings and standard evaluation metrics (F1 for LoCoMo, exact match for MediQ).

5.1 Main Results

Table 1 shows our hybrid approach consistently outperforms local-only baselines by up to 27.6 percentage points while approaching or exceeding remote-only performance. Notably, our framework with Llama-3.2-1B achieves F1=87.7 on LoCoMo, surpassing GPT-4o (80.6) by 7.1 points. This counterintuitive result demonstrates that decomposing tasks between untrusted powerful LLMs and trusted local models improves performance through structured test-time computation. On MediQ, improvements are smaller due to domain-specific challenges, but our approach still provides substantial gains over local-only baselines while maintaining complete privacy of medical records.

Table 1: Main Results: Our hybrid framework outperforms both local-only and remote-only baselines.

Method	**Privacy**	**LoCoMo F1**	**MediQ EM**
Local-only (Llama-1B)	✓	64.6	40.3
Local-only (Llama-3B)	✓	69.4	43.7
Remote-only (GPT-4o)	×	80.6	89.2
Remote-only (Gemini-1.5-Pro)	×	84.3	87.5
Ours (Hybrid)	✓	**87.7**	**67.9**
Improvement over Local		**+23.1**	**+27.6**
Improvement over GPT-4o		**+7.1**	**–**

5.2 Ablation Study

Table 2 isolates the contributions of sub-query generation and chain-of-thought reasoning. Delegating sub-query generation to GPT-4o doubles retrieval performance (Recall@5: 21.8→44.1 on LoCoMo), while GPT-4o-generated reasoning prompts improve final answer quality. Both components are essential for optimal performance. **Database Performance:** Our encrypted vector database maintains >99% search accuracy across LoCoMo, Deep1B, and LAION benchmarks with 5.8× storage overhead and sub-second latency on million-scale collections. Network communication becomes the primary bottleneck rather than homomorphic computation. The results demonstrate that

Table 2: Ablation Study: Both sub-query generation and CoT reasoning contribute to performance.

Sub-query Source	CoT Source	Recall@5	LoCoMo F1
Llama-1B	Llama-1B	21.8	82.0
GPT-4o	Llama-1B	44.1	85.4
GPT-4o	GPT-4o	44.1	**87.7**
Ground Truth	GPT-4o	100.0	89.3

our framework enables effective collaboration between untrusted powerful models and trusted local models, achieving better performance than either approach alone while preserving complete privacy of personal data.

6 Related Work

Private Inference via Encryption. Early approaches combined homomorphic encryption with neural networks [15], achieving privacy with $10^3\times$ computational overhead. Recent systems like MPCFormer [26], PermLLM [51], and PUMA [12] extend these to Transformers but require seconds per token. Cloud providers remain reluctant to adopt these approaches due to computational costs and complex key management.

Input Sanitization Methods. Complementary approaches sanitize prompts before transmission. PREEMPT [11] replaces sensitive spans with placeholders, while PAPILLON [40] divides processing between local and external LLMs. These methods require task-specific engineering and often sacrifice accuracy when critical context is removed [47].

Chain-of-Thought and Task Decomposition. CoT prompting improves LLM reasoning through step-by-step solutions [46, 23]. Model cascades like FrugalGPT [9] route queries between different-sized models using confidence estimators. Multi-model frameworks like Socratic Models [48] divide tasks between planners and executors but assume the central model has full access to private data.

RAG and Agentic Workflows. Modern systems embed LLMs within persistent datastores for personalized assistance, from research prototypes like Generative Agents [35] to commercial deployments like ChatGPT Memory [34]. However, these systems typically assume trustworthy datastores, ignoring privacy risks from extraction attacks [3].

Our work is the first to combine agentic RAG with encrypted local retrieval, enabling powerful model collaboration while maintaining strict privacy guarantees through cryptographic protection rather than data minimization or sanitization.

7 Conclusion and Discussion

We introduced a four-stage, privacy-preserving framework that uniquely partitions tasks between untrusted powerful LLMs and trusted lightweight local models. Our key innovations—Socratic Chain-of-Thought Reasoning and Homomorphically Encrypted Vector Database—enable secure collaboration without exposing private data. Our approach not only preserves privacy but actually improves performance, with our local lightweight model outperforming even GPT-4o on long-context QA tasks. This counter-intuitive result demonstrates the power of additional test-time computation when properly structured through our chain-of-thought decomposition. Meanwhile, our encrypted vector database achieves sub-second latency on million-scale collections with negligible accuracy loss compared to plaintext search.

Future work should address extending our approach to tasks resistant to clean decomposition, developing dynamic sensitivity classification for mixed public-private content, and scaling encrypted retrieval to billion-scale collections. These advances will further expand applications that can benefit from powerful models without surrendering personal data.

References

[1] Anthropic. Claude 3.5 sonnet, 2024. URL https://www.anthropic.com/news/claude-3-5-sonnet. Accessed: 2025-01-29.

[2] Artem Babenko and Victor Lempitsky. Efficient indexing of billion-scale datasets of deep descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[3] Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Röttger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. Safety-tuned llamas: Lessons from improving the safety of large language models that follow instructions. *arXiv preprint arXiv:2309.07875*, 2023. URL https://arxiv.org/abs/2309.07875.

[4] Fabian Boemer, Sejun Kim, Gelila Seifu, Fillipe DM de Souza, and Vinodh Gopal. Intel hexl: accelerating homomorphic encryption with intel avx512-ifma52. In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2021.

[5] Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher. Quantum security analysis of aes. *IACR Transactions on Symmetric Cryptology*, 2019(2):55–93, 2019.

[6] Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 587–617. Springer, 2021.

[7] Dawn M Cappelli, Andrew P Moore, and Randall F Trzeciak. *The CERT guide to insider threats: how to prevent, detect, and respond to information technology crimes*. Addison-Wesley, 2012.

[8] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) lwe ciphertexts. In *International conference on applied cryptography and network security*, pages 460–479. Springer, 2021.

[9] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023. URL https://arxiv.org/abs/2305.05176.

[10] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, 2017.

[11] Amrita Roy Chowdhury, David Glukhov, Divyam Anshumaan, Prasad Chalasani, Nicolas Papernot, Somesh Jha, and Mihir Bellare. Preempt: Sanitizing sensitive prompts for llms. *arXiv preprint arXiv:2504.05147*, 2025.

[12] Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, and Wenguang Chen. PUMA: Secure inference of LLaMA-7B in five minutes. *CoRR*, abs/2307.12533, 2023. doi: 10.48550/arXiv.2307.12533.

[13] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[14] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[15] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 201–210, 2016.

[16] Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in helib. In *CRYPTO*, 2018.

[17] Jeffrey Hunker and Christian W Probst. Insiders and insider threats-an overview of definitions and mitigation techniques. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 2(1):4–27, 2011.

[18] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[19] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Technical report, Lockheed Martin Corporation, 2011.

[20] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

[21] Changyue Jiang, Xudong Pan, Geng Hong, Chenfu Bao, and Min Yang. Rag-thief: Scalable extraction of private data from retrieval-augmented generation applications with agent-based attacks. *arXiv preprint arXiv:2411.14110*, 2024.

[22] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[23] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022. URL `https://arxiv.org/abs/2205.11916`.

[24] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[25] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems 33*, pages 9459–9474, 2020.

[26] Dacheng Li, Rulin Shao, Hongyi Wang, Han Guo, Eric P. Xing, and Hao Zhang. MPCFormer: Fast, performant and private transformer inference with MPC. *CoRR*, abs/2211.01452, 2022. doi: 10.48550/arXiv.2211.01452.

[27] Shuyue Stella Li, Vidhisha Balachandran, Shangbin Feng, Jonathan Ilgen, Emma Pierson, Pang Wei Koh, and Yulia Tsvetkov. Mediq: Question-asking llms for adaptive and reliable medical reasoning. *arXiv preprint arXiv:2406.00922*, 2024.

[28] Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. How to train your dragon: Diverse augmentation towards generalizable dense retrieval. *arXiv preprint arXiv:2302.07452*, 2023.

[29] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, and Kejuan Yang. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

[30] Yuxuan Liu et al. A review on edge large language models: Design, execution, and optimization. *ACM Computing Surveys*, 1(1):1–36, 2025.

[31] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

[32] Samir Jordan Menon and David J. Wu. Spiral: Fast, high-rate single-server PIR via FHE composition. Cryptology ePrint Archive, Paper 2022/368, 2022.

[33] Niloofar Mireshghallah, Hyunwoo Kim, Xuhui Zhou, Yulia Tsvetkov, Maarten Sap, Reza Shokri, and Yejin Choi. Can llms keep a secret? testing privacy implications of language models via contextual integrity theory. *arXiv preprint arXiv:2310.17884*, 2023.

[34] OpenAI. Memory and new controls for chatgpt. `https://openai.com/index/memory-and-new-controls-for-chatgpt/`, February 2024. Accessed 16 May 2025.

[35] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023. URL `https://arxiv.org/abs/2304.03442`.

[36] Jianing Qiu, Kyle Lam, Guohao Li, Amish Acharya, Tien Yin Wong, Ara Darzi, Wu Yuan, and Eric J Topol. Llm-based agentic systems in medicine and healthcare. *Nature Machine Intelligence*, 6(12):1418–1420, 2024.

[37] Xuanle Ren, Zhaohui Chen, Zhen Gu, Yanheng Lu, Ruiguang Zhong, Wen-Jie Lu, Jiansong Zhang, Yichi Zhang, Hanghang Wu, Xiaofu Zheng, Heng Liu, Tingqiang Chu, Cheng Hong, Changzheng Wei, Dimin Niu, and Yuan Xie. Cham: A customized homomorphic encryption accelerator for fast matrix-vector product. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2023. doi: 10.1109/DAC56929.2023.10247696.

[38] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[39] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 2022.

[40] Li Siyan, Vethavikashini Chithrra Raghuram, Omar Khattab, Julia Hirschberg, and Zhou Yu. Papillon: Privacy preservation from internet-based and local language model ensembles. *arXiv preprint arXiv:2410.17127*, 2024.

[41] Jaeyoon Song, Zahra Ashktorab, and Thomas W Malone. Togedule: Scheduling meetings with large language models and adaptive representations of group availability. *arXiv preprint arXiv:2505.01000*, 2025.

[42] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings. *arXiv preprint arXiv:2212.09741*, 2022.

[43] Fireworks Team. Fireworks api documentation, 2025. Available at `https://docs.fireworks.ai/`.

[44] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[45] David Wang et al. The pros and cons of using large language models (llms) in the cloud vs. running llms locally. *DataCamp*, 2024.

[46] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022. URL `https://arxiv.org/abs/2201.11903`.

[47] Rui Xin, Niloofar Mireshghallah, Shuyue Stella Li, Michael Duan, Hyunwoo Kim, Yejin Choi, Yulia Tsvetkov, Sewoong Oh, and Pang Wei Koh. A false sense of privacy: Evaluating textual data sanitization beyond surface-level privacy leakage. *arXiv preprint arXiv:2504.21035*, 2025.

[48] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022. URL `https://arxiv.org/abs/2204.00598`.

[49] Shenglai Zeng, Jiankun Zhang, Pengfei He, Yiding Liu, Yue Xing, Han Xu, Jie Ren, Yi Chang, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. The good and the bad: Exploring privacy issues in retrieval-augmented generation (RAG). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4505–4524, 2024.

[50] Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, et al. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag). *arXiv preprint arXiv:2402.16893*, 2024.

[51] Fei Zheng, Chaochao Chen, Zhongxuan Han, and Xiaolin Zheng. PermLLM: Private inference of large language models within 3 seconds under WAN. *CoRR*, abs/2405.18744, 2024. doi: 10.48550/arXiv.2405.18744.

[52] Jinhao Zhu, Liana Patel, Matei Zaharia, and Raluca Ada Popa. Compass: Encrypted semantic search with high accuracy. Cryptology ePrint Archive, Paper 2024/1255, 2024.

A Homomorphic Encryption based Inner Product

A.1 Secure Inner Product, Algorithms and Optimizations

We specify the detailed algorithms as follows. Algorithms 1 and 2 describe the precomputations for the query and key, respectively. Algorithm 3 describes the score computation algorithm starting from the precomputed query and cache ciphertexts.

Optimizations Summary. We summarize the optimizations mentioned in the previous subsection and discuss some additional optimizations.

- **Batching and Caching**: We write the homomorphic inner product equation. This separates the precomputations for query and key, which are denoted as `Decompose` and `Cache`, respectively. This reduces the number of automorphisms from $d\log(r)$ to $r-1$.
- **Butterfly Decomposition**: The key side precomputation is significant as it involves $O(r^2)$ polynomial additions. We leverage the butterfly decomposition to reduce the complexity from $r(r-1)$ to $r\log(r)$.
- **Seeding and MLWE**: In order to improve the storage size, we use Module LWE (MLWE) [24] and Extendable Output-format Function (XOF) with a public seed. This reduces ciphertext size from $2d$ (i.e. two $\mathcal{R}_{q,d}$ elements) to r (i.e. one $\mathcal{R}_{q,r}$ element and a 128-bit public seed).
- **Remove the leading term** r: We use the optimization technique introduced in [8] that evaluates the trace without the leading term r, thereby improving the precision. This technique is applied for Line 2 of Algorithm 1 and Line 3 of Algorithm 2.
- **Hoisting** [16]: We adapt the hoisting technique that lazily computes the homomorphic operations to improve efficiency. Our adaptaion is similar to the double hoisting algorithm in [6]. Hoisting appears in the following instances.
  - Line 3 of Algorithm 1: For each index $0 \le i < s$, $\texttt{ModUp}(a_i)$ is computed only once.
  - Line 5,6 of Algorithm 1, Line 13,14 of Algorithm 2: We `ModDown` after summation, reducing the number of `modDown` to r per each j.
- **Reducing NTT dimension**: In Line 3,5,6 of Algorithm 1, we utilize dimension r NTT instead of dimension d NTT, reducing the complexity by a factor of $\log(d)/\log(r)$. This is possible because each $\hat{a}_i$ is sparsely embedded into the larger ring $\mathcal{R}_{q,d}$.

Algorithm 1 `Decompose`

Require: Query (seeded) MLWE ciphertext (b, ρ) that encrypts $\mathrm{q} \in \mathcal{R}_{q,r}$ via the secret key $\mathbf{s} = (s_u)_{0\le u<s} \in \mathcal{R}^s_{q,r}$. Here $b \in \mathcal{R}_{q,r}$ and ρ is a 128-bit seed string. $\texttt{swk}_j = (\texttt{swk}_{j,u})_{0\le u<s} \in (\mathcal{R}^2_{qp,d})^s$ are the RLWE switching keys where $\texttt{swk}_{j,u}$ switches from $\tilde{s_u}$ to $\varphi_j^{-1}(s')$ where $s' \in \mathcal{R}_{*,d}$ is the target RLWE secret key. Here `GenA` generates the a-part of the MLWE ciphertext from the 128-bit seed ρ, and `ModUp` and `ModDown` are the typical homomorphic base conversions from q to qp and from qp to q.

Ensure: RLWE ciphertexts $(ct_j)_{0\le j<r}$ that encrypt $\left(\varphi_j(r^{-1}\cdot \mathrm{q})\right)_{0\le j<r}$, i.e. polynomial of degree d in $\mathcal{R}_q$ with X^{2j+1} automorphism operations for $0 \le j < r$.

1: $\mathbf{a} = (a_u)_{0\le u<s} \in \mathcal{R}^s_{q,r} \leftarrow \texttt{GenA}(\rho)$
2: $(b, \mathbf{a}) \leftarrow r^{-1}\cdot(b, \mathbf{a}) \bmod q$
3: $\hat{\mathbf{a}} = (\hat{a}_u)_{0\le u<s} \in \mathcal{R}^s_{qp,r} \leftarrow (\texttt{ModUp}(a_u))_{0\le u<s}$
4: **for** $j = 0$ to $r-1$ **do**
5: $\quad ct_j \in \mathcal{R}^2_{qp,d} \leftarrow \sum_{u=0}^{s-1}(\hat{a}_i \cdot \texttt{swk}_{j,u})$
6: $\quad ct_j \leftarrow \texttt{ModDown}(ct_j)$
7: $\quad ct_j \leftarrow \varphi_j(ct_j + (\tilde{b} \in \mathcal{R}_{q,d},\ 0))$
8: **end for**
9: **return** $(ct_j)_{0\le j<r}$

Algorithm 2 `Cache`

Require: Key (seeded) MLWE ciphertexts (b_i, ρ_i) that encrypts $k_i \in \mathcal{R}_{q,r}$ via the secret key $\mathbf{s} = (s_u)_{0 \le u < s} \in \mathcal{R}_{q,r}^s$, for each $0 \le i < d$. Here $b_i \in \mathcal{R}_{q,r}$ and ρ_i is a 128-bit seed string. $\texttt{swk}_j = (\texttt{swk}_{j,u})_{0 \le u < s} \in (\mathcal{R}_{qp,d}^2)^s$ are the RLWE switching keys where $\texttt{swk}_{j,u}$ switches from $\varphi_j(\tilde{s}_i)$ to s' where $s' \in \mathcal{R}_{*,d}$ is the target RLWE secret key. Here `GenA` generates the a-part of the MLWE ciphertext from the 128-bit seed ρ, and `ModUp` and `ModDown` are the typical homomorphic base conversions from q to qp and vice versa, respectively. Let $\mathbf{B} \in \mathcal{R}_{q,d}^{r \times r}$ be the matrix.

Ensure: RLWE ciphertexts $(ct_j''')_{0 \le j < r} \in (\mathcal{R}_{q,d}^2)^r$ that encrypt $\left(\sum_{i=0}^{d-1} \varphi_j(\tilde{k}_i) X^i\right)_{0 \le j < r}$.

for $i = 0$ to $d - 1$ **do**
  $\mathbf{a}_i = (a_{i,u})_{0 \le u < s} \in \mathcal{R}_{q,r}^s \leftarrow \texttt{GenA}(\rho_i)$
  $(b_i, \mathbf{a}_i) \leftarrow r^{-1} \cdot (b_i, \mathbf{a}_i) \bmod q$
end for
for $j = 0$ to $r - 1$ **do**
  $(b'_j, \mathbf{a}'_j) \in \mathcal{R}_{q,d}^{s+1} \leftarrow \left(\sum_{v=0}^{s-1} \tilde{b}_{v+sj} \cdot X^v, \left(\sum_{v=0}^{s-1} \tilde{a}_{(v+sj),u} \cdot X^v\right)_{0 \le u < s}\right)$
end for
$\mathbf{ct}' \in (\mathcal{R}_{q,d}^{s+1})^r \leftarrow (b'_j, \mathbf{a}'_j)_{0 \le j < r}$
$\mathbf{ct}' \in (\mathcal{R}_{q,d}^{s+1})^r \leftarrow \mathbf{B} \cdot \mathbf{ct}'$
for $j = 0$ to $r - 1$ **do**
  $ct''_j = (b''_j, \mathbf{a}''_j) \in \mathcal{R}_{q,d} \times \mathcal{R}_{q,d}^s \leftarrow \varphi_{j,r}(\mathbf{ct}'[j])$
  $\hat{\mathbf{a}}''_j = (\hat{a}''_{j,u})_{0 \le u < s} \in \mathcal{R}_{qp,d}^s \leftarrow \texttt{ModUp}(\mathbf{a}''_j)$
  $ct'''_j \in \mathcal{R}_{qp,d}^2 \leftarrow \sum_{u=0}^{s-1} (\hat{a}''_{j,u} \cdot \texttt{swk}_{j,u})$
  $ct'''_j \in \mathcal{R}_{q,d}^2 \leftarrow \texttt{ModDown}(ct'''_j)$
  $ct'''_j \leftarrow ct'''_j + (b''_j \in \mathcal{R}_{q,d},\ 0)$
  $ct'''_j \leftarrow r \cdot ct'''_j \bmod q$
end for
return $(ct'''_j)_{0 \le j < r}$

Algorithm 3 `Score`

Require: `Decomposed` query ciphertexts $\mathbf{ct}_q \in (\mathcal{R}_{q,d}^2)^r$, `Cached` key ciphertexts $\mathbf{ct}_k \in (\mathcal{R}_{q,d}^2)^r$.

Ensure: A RLWE ciphertext ct_{out} encrypting the resulting score polynomial $\sum_{j=0}^{d-1} \sigma_j X^j$.

$ct_{out} \leftarrow \texttt{Relin}(\sum_{i=0}^{r-1} \mathbf{ct}_q[i] \otimes \mathbf{ct}_k[i])$
return ct_{out}

A.2 Private Information Retrieval

We extend our Secure Inner Product method to support Private Information Retrieval (PIR). Similar to SPIRAL [32], we treat the database as a matrix. The protocol requires the client to send two encrypted queries: one selecting the target row and the other selecting the target column, each containing a one hot vector at the corresponding index. The server then performs PIR through two sequential applications of the Secure Inner Product protocol. However, naively applying the Secure Inner Product protocol in this PIR context introduces a cache invalidation issue. Specifically, while the standalone Secure Inner Product scenario only requires refreshing the cache corresponding to the updated index, PIR necessitates refreshing the entire cache whenever the database changes. This occurs because the output from the first stage acts as the key for the second stage. To address this, we modify our protocol by applying the inverse butterfly operation—originally intended for use on the key—to the decomposed query instead.

In our experimental setting using a Fast network (see Section C), the modified PIR protocol achieves an end-to-end retrieval latency of under 700 ms for databases consisting of 2^{20} records, each sized at 1 KiB. Consequently, we demonstrate that our approach efficiently supports a secure vector database

of 1 GiB containing 1 million records with 96 dimensions each, achieving an end-to-end latency below 1 second.

B Experimental Setup

B.1 Socratic Chain-of-Thought Reasoning

We empirically evaluate the effectiveness of our reasoning framework in addressing the computational limitations of local models. Experiments are conducted on two QA-focused benchmarks: LoCoMo, which simulates personal assistant scenarios, and MediQ, which simulates medical consultation scenarios. Both tasks require retrieving relevant private user data and performing complex reasoning to arrive at a final answer. We compare our framework against two categories of baselines: Golden Baselines assume no privacy constraints, allowing private data to be directly passed to remote models. We use GPT-4o (R1), Gemini-1.5-Pro (R2), and Claude-3.5-Sonnet (R3), which cannot be run locally but offer strong reasoning capabilities. Local-only Baselines assume strong privacy constraints, requiring the entire inference process to be carried out by local models. We use Llama-3.2-1B (L1), Llama-3.2-3B (L2), and Llama-3.1-8B (L3), which are lightweight enough for local execution but less capable in complex reasoning tasks. The goal of our reasoning framework is to improve the performance of local-only baselines by leveraging model collaboration and delegated reasoning, aiming to approach the performance of the golden baselines.

B.2 Homomorphically Encrypted Vector Database

We examine whether vector search can be performed accurately and efficiently over encrypted data using homomorphic encryption. Our goal is to match the quality and latency of plaintext vector search while ensuring that both queries and database contents remain private. The encrypted vector database is implemented using HEXL [4] and evaluated in in the same Google Cloud Platform configuration used by Compass [52] for a fair comparison: an n2-standard-8 instance (8 vCPUs @ 2.8 GHz, 32 GB RAM) as the client and an n2-highmem-64 instance (64 vCPUs @ 2.8 GHz, 512 GB RAM) as the server, co-located in the same region/zone. Using Linux Traffic Control, we emulate two network regimes: Fast (3 Gbps, 1 ms Round Trip Time (RTT)) and Slow (400 Mbps, 80 ms RTT) to isolate the impact of bandwidth and latency. We use 10k query vectors and 1M key vectors from Deep1B (96D) and LAION (512D), as well as the entire LoCoMo dataset (768D). For search accuracy, we report mean/max inner product error, MRR@10, and 1-Recall@k. For latency, we measure end-to-end CPU runtime. All speed measurements assume that both the query and the keys are ciphertexts and employ parameters that satisfy IND-CPA 128-bit security. To evaluate storage, we analyze ciphertext overhead and apply packing optimizations.

B.3 Hyperparameter Selection

To evaluate Socratic Chain-of-Thought Reasoning, we set the temperature of all language models to zero to ensure reproducibility. We use top-k retrieval with reranking based on vector similarity scores. We set k to 5 for LoCoMo and 20 for MediQ, as the maximum number of ground truth retrievals varies across datasets.

B.4 Model Selection

We employ DRAGON [28] as the retriever because it outperforms other candidates, such as DPR [22], Contriever [20], and Instructor [42], on our chosen datasets. It represents data as 768-dimensional vectors, and the inner product between two vectors is used to compute the similarity score. For the remote models, we use GPT-4o (R1) [18], Gemini-1.5-Pro (R2) [44], and Claude-3.5-Sonnet (R3) [1], representing the most powerful closed API language models currently available. These models are assumed to run in a public cloud environment. For the local models, we select Llama-3.2-1B (L1), Llama-3.2-3B (L2), and Llama-3.1-8B (L3) [13], which are lightweight enough to be deployed on edge devices. These models reflect realistic constraints for privacy-preserving, on-device inference. This selection enables a clear evaluation of our framework, balancing reasoning capability with privacy constraints.

B.5 Benchmark Selection

We report the performance of Socratic Chain-of-Thought Reasoning on two benchmarks. The first, LoCoMo [31], is a benchmark designed to test language models in long-term dialogues. It simulates an everyday personal assistance scenario, where personal information is gradually accumulated in a vector database through extended observation. On LoCoMo, we evaluate (1) the remote models's impact on retrieval using Recall@5 and (2) its enhancement of response quality through improved response generation, measured by the F1 score. We use only the single-hop QA and multi-hop QA datasets out of the total five datasets in LoCoMo, as these are the only datasets suitable for our scenario. The second benchmark, MediQ [27], presents a more specialized scenario focused on medical consultation, where privacy risks are directly at odds with the need for access to a patient's personal context. MediQ is a multiple-choice question-answering dataset, so we evaluate generation accuracy using the exact match metric. Since MediQ lacks retrieval annotations, we do not report retrieval metric for this benchmark.

We report the performance of the homomorphically encrypted vector database on standard retrieval benchmarks. To assess the scalability of encrypted storage and search, we selected a sufficiently large dataset. We used the top 10k query vectors and 1M key vectors from Deep1B [2] and LAION [39], represented as 96-dimensional and 512-dimensional vectors respectively. For LoCoMo [31], we used the entire dataset, which consists of 1,742 query vectors and 4,972 key vectors, each represented as a 768-dimensional vector.

B.6 Metric Selection

For the Socratic Chain-of-Thought Reasoning, we focus on measuring the quality of the generated answers. On the LoCoMo benchmark, we report the F1 score, which captures token-level overlap between generated and ground-truth responses in long-context dialogues. On the MediQ benchmark, we report exact match accuracy, as the task involves multiple-choice question answering and requires strict correctness. These metrics enable us to quantify the impact of delegating complex reasoning to powerful remote models while keeping sensitive data within a trusted zone.

For the homomorphically encrypted vector database, we evaluate both search accuracy and latency. To assess search accuracy, we compute the mean error and maximum error between the inner product similarity scores produced by encrypted and plaintext searches. Additionally, we report 1-Recall@1 and 1-Recall@5, which represent the proportion of queries for which the top-1 result from the plaintext database is not recovered in the top-1 or top-5 encrypted results. Lower values for these metrics indicate higher retrieval consistency under encryption. To evaluate latency, we measure the average response time of encrypted search queries. All metrics are reported separately for plaintext and ciphertext queries.

C Compute Resources

For Socratic Chain-of-Thought Reasoning, all experiments were conducted using a single NVIDIA A100 GPU. Language models from the Llama family were accessed via the Fireworks API [43], while other closed API models, including those from OpenAI, Gemini, and Claude, were accessed through their respective APIs. Our homomorphically encrypted vector database was implemented using HEXL [4] and evaluated under the same Google Cloud Platform configuration used by Compass [52] to ensure a fair comparison: an n2-standard-8 instance (8 vCPUs @ 2.8 GHz, 32 GB RAM) was used as the client, and an n2-highmem-64 instance (64 vCPUs @ 2.8 GHz, 512 GB RAM) was used as the server, both co-located in the same region and zone. To emulate realistic networking conditions, we used Linux Traffic Control to simulate two environments: **Fast** (3 Gbps bandwidth, 1 ms round-trip time and **Slow** (400 Mbps bandwidth, 80 ms round-trip time). The following commands were used to apply these network configurations to the server.

Fast Network

```
tc qdisc add dev ens4 root netem delay 1ms
tc qdisc add dev ens4 root handle 1: htb default 30
tc class add dev ens4 parent 1: classid 1:1 htb rate 3096mbps
tc class add dev ens4 parent 1: classid 1:2 htb rate 3096mbps
```

```
tc filter add dev ens4 protocol ip parent 1:0 prio 1 u32 \
match ip dst $CLIENT_IP flowid 1:1
tc filter add dev ens4 protocol ip parent 1:0 prio 1 u32 \
match ip src $CLIENT_IP flowid 1:2
```

Slow Network

```
tc qdisc add dev ens4 root netem delay 80ms
tc qdisc add dev ens4 root handle 1: htb default 30
tc class add dev ens4 parent 1: classid 1:1 htb rate 400mbps
tc class add dev ens4 parent 1: classid 1:2 htb rate 400mbps
tc filter add dev ens4 protocol ip parent 1:0 prio 1 u32 \
match ip dst $CLIENT_IP flowid 1:1
tc filter add dev ens4 protocol ip parent 1:0 prio 1 u32 \
match ip src $CLIENT_IP flowid 1:2
```

D Qualitative Analysis

We present qualitative examples from the LoCoMo and MediQ benchmarks to illustrate how our system improves response quality under strict privacy constraints. By delegating sub-query generation and chain-of-thought reasoning to a powerful remote model, and executing final response generation locally, our framework ensures that sensitive data never leaves the trusted zone while still benefiting from advanced reasoning capabilities.

D.1 LoCoMo

User Query. *"What motivated Caroline to pursue counseling?"*

This query requires linking the user's past personal experiences to her career decisions, as this information is often buried in long conversational histories.

Sub-Query Generation by Remote Model. The remote model generated sub-queries such as: *"Has Caroline discussed any impactful personal experiences related to her career?" "Did she mention an interest in counseling in past conversations?"*

These sub-queries were embedded on the local client and used to search the homomorphically encrypted vector database.

Encrypted Search from Private Records. The search retrieved a key statement: *"My own journey and the support I got made a huge difference... I saw how counseling and support groups improved my life."*

Chain-of-Thought Reasoning from Remote Model. The model suggested this reasoning guideline: *"When personal growth or transformation is attributed to support or counseling, infer a connection between that experience and a career motivation to help others."*

Response Generation by Local Model. Using the retrieved memory and the reasoning instruction, the local model generated the following answer: *"Caroline was motivated to pursue counseling because of her own journey and the support she received, particularly through counseling and support groups."*

D.2 MediQ

User Query. *"I've been feeling more forgetful lately and have started falling more often. What should I do?"*

This query suggests a combination of cognitive and physical decline, potentially indicating an underlying neurological issue. Proper assessment requires integration of personal medical context and symptom history.

Sub-Query Generation by Remote Model. The remote model generated targeted follow-up questions, including: *"Is there any record of short-term memory impairment?" "Have the falls*

become more frequent or severe over time?" "Are there other neurological symptoms noted in the history?"

Encrypted Search from Private Records. These sub-queries were executed on encrypted medical records, retrieving relevant notes such as: *"I couldn't remember any of the five things the doctor asked me to recall after ten minutes." "I've been falling more often lately, and it feels like it's getting worse."*

Chain-of-Thought Reasoning from Remote Model. The remote model provided the following reasoning instruction to the local model: *"When both progressive memory loss and increased frequency of falls are reported, evaluate for possible neurodegenerative conditions and recommend medical assessment."*

Response Generation by Local Model. Based on the retrieved data and reasoning instruction, the local model generated the following concise response: *"Parkinson's disease."*

These examples demonstrate that our framework enables local models to generate informed, context-sensitive responses by leveraging powerful remote models for high-level reasoning. Throughout the process, sensitive user data remains local, ensuring strong privacy guarantees while maintaining or even improving response quality.

E Prompt Templates

For sub-query generation in both the baselines and Socratic Chain-of-Thought Reasoning, we used the prompt shown in Figure 2. For response generation in the baselines, the prompt in Figure 3 was used. For Socratic Chain-of-Thought Reasoning, chain-of-thought generation was performed using the prompt in Figure 4, and response generation used the prompt in Figure 5. The prompts include substitution keys, which are described in Table 3.

Key	Description	Illustrative Example
{user_input}	User input	`I have a fever and a cough. What disease do I have?`

{options}	Multiple-choice option. Formatted as bulleted list. For open ended questions, this is replaced with `Empty` instead.	`- Common cold`
		`- Flu`
		`- Strep throat`

{personal_context}	List of retrieved personal contexts in descending order of importance, one item on each line.	`In January 30th, user consumed a half gallon of ice cream.`
		`User enjoys cold drink, even in winter.`
		`User spends most of the time in their place alone.`

{personal_context_json}	List of retrieved personal contexts in descending order of importance, as JSON-formatted array of strings.	`[`
		`  "In January 30th, user consumed a half gallon of ice cream.",`
		`  "User enjoys cold drink, even in winter.",`
		`  "User spends most of the time in their place alone."`
		`]`

{generated_reasoning}	The output of reasoning generation step.	(omitted)

Table 3: Substitutions for our prompts. Whenever the listed substitution keys appear on our prompt template, they are substituted into the actual values as described on the right side of the table.

```
You are a sub-query generator.

1.  You are given a query and a list of possible options.
2.  Your task is to generate 3 to 5 sub-queries that help retrieve
personal context relevant to answering the query.
3.  Each sub-query should be answerable based on the user's personal
context.
4.  Ensure the sub-queries cover different aspects or angles of the
query.
5.  If the options text says 'Empty,' it means no options are
provided.

Please output the sub-queries one sub-query each line, in the
following format:
"Sub-query 1 here"
"Sub-query 2 here"
"Sub-query 3 here"

Example 1)

## Query
I have a fever and a cough.  What disease do I have?

## Options
Common cold
Flu
Strep throat

### Sub-queries
"Have user visited any countries in Africa recently?"
"Have user eat any cold food recently?"
"Have user been in contact with anyone who has a COVID-19 recently?"

Test Input)

### Query
{user_input}

### Options
{options}

### Sub-queries
```

Figure 2: Prompt used for sub-query generation in both the baselines and the socratic chain-of-thought reasoning.

```
You are a question answering model.

1.  You are given a personal context, a query, and a list of
possible options.
2.  Your task is to generate an answer to the query based on the
user's personal context.
3.  You should generate an answer to the query by referring to the
personal context where relevant.
4.  If the options text says 'Empty,' it means no options are
provided.
5.  If the options are not empty, simply output one of the answers
listed in the options without any additional explanation.
6.  Never output any other explanation.  Just output the answer.
7.  If option follows a format like '[A] something', then output
something as the answer instead of A.

Test Input)


### Personal Context
{personal_context}

### Question
{user_input}

### Options
{options}

### Answer
```

Figure 3: Prompt used for response generation in the baselines.

```
Your task is to provide good reasoning guide for students.

You are a chain-of-thought generator.
1.  You are given a query and a list of possible options.
2.  Your task is to provide a step-by-step reasoning guide to help a
student answer the query.
3.  The reasoning guide should clearly show your reasoning process
so that the student can easily apply it to their query.
4.  Analyze the query and write a reasoning guide for the student to
follow.
5.  If there is a lack of information relevant to the query, you
must identify the missing elements as "VARIABLES" and write the
guide on a case-by-case basis.
6.  If the options text says 'Empty,' it means no options are
provided.

Test Input)

### Query
{user_input}

### Options
{options}

### Chain-of-Thought
```

Figure 4: Prompt used for chain-of-thought generation in the socratic chain-of-thought reasoning.

```
You are a question answering model.

1.  Your task is to answer the query based on the teacher's
chain-of-thought decision guide, using additional personal context.
2.  Read the chain-of-thought decision guide carefully.
3.  If the decision guide contains "VARIABLES" that may affect
the outcome, extract them and determine their values based on the
personal context.
4.  Then, follow the decision guide and apply the extracted
variables appropriately to derive the final answer.
5.  The final answer must be preceded by '### Answer', and your
response must end immediately after the answer.
6.  If the options text says 'Empty,' it means no options are
provided.
7.  If the options are not empty, simply output one of the answers
listed in the options without any additional explanation.
8.  Never output any other explanation.  Just output the answer.
9.  If option follows a format like '[A] something', then output
something as the answer instead of A.

### Personal Context
{personal_context_json}

### Chain-of-Thought
{cot}

### Query
{user_input}

### Options
{options}

### Answer
```

Figure 5: Prompt used for response generation in the socratic chain-of-thought reasoning.

F Additional MediQ Analysis

The Remote-Only Baseline with Socratic Chain-of-Thought Reasoning performs worse than the standard Remote-Only Baseline on MediQ. To understand the cause of this drop, we conducted a detailed qualitative analysis of the model's inputs and outputs. As a result, we found that R1 (GPT-4o), when generating chain-of-thought reasoning, often included the most likely answer without considering the user's personal context. As a result, L1 (Llama-3.2-1B) became strongly biased toward this uncontextualized answer and also ignored the user's personal context. To address this issue, we added explicit rules to the prompt—shown in Figure 6—to reduce this bias and re-ran the experiment under this setup only. With this adjustment, performance improved from 67.3 to 77.0, indicating that the bias was partially mitigated.

```
Your task is to provide good reasoning guide for students.

You are a chain-of-thought generator.
1.  You are given a query and a list of possible options.
2.  Your task is to provide a step-by-step reasoning guide to help a
student answer the query.
3.  The reasoning guide should clearly show your reasoning process
so that the student can easily apply it to their query.
4.  Analyze the query and write a reasoning guide for the student to
follow.
5.  The student may have less domain knowledge than you, but they
have more context about the situation.
6.  If there is a lack of information relevant to the query, you
must identify the missing elements as "VARIABLES" and write the
guide on a case-by-case basis.
7.  Since you don't have full context about the situation, your goal
is not to choose a final answer but to present a set of possible
answers along with the reasoning steps that could lead to each one.
8.  If the options text says 'Empty,' it means no options are
provided.

Test Input)

### Query
{user_input}

### Options
{options}

### Chain-of-Thought
```

Figure 6: Prompt used for chain-of-thought generation in the additional MediQ analysis.