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

Yubeen Bae[1*] **Minchan Kim**[1*] **Jaejin Lee**[1*] **Sangbum Kim**[1] **Jaehyung Kim**[2]
Yejin Choi[3] **Niloofar Mireshghallah**[4]
[1]Seoul National University [2]Stanford University [3]NVIDIA [4]University of Washington
{lights0320, kjkk0502, jaejin.lee}@snu.ac.kr

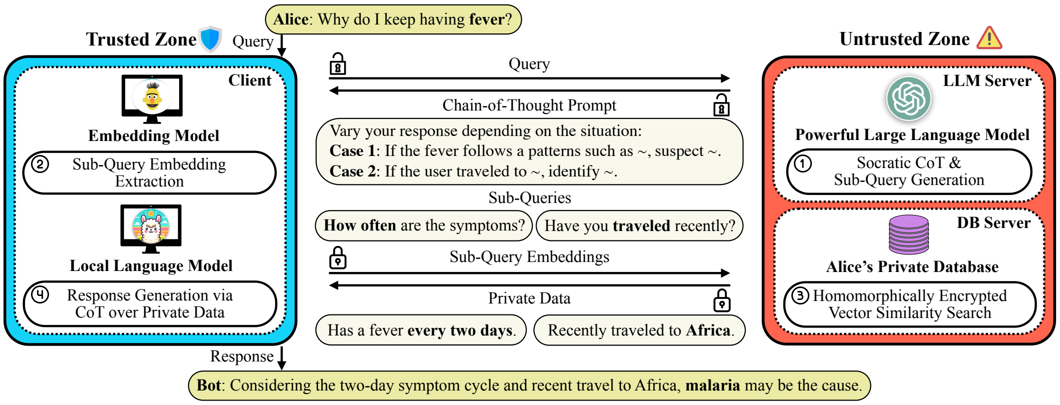
Alice: Why do I keep having fever?
Trusted Zone
Query
Client
Embedding Model
② Sub-Query Embedding Extraction
Local Language Model
④ Response Generation via CoT over Private Data
Response
Query
Chain-of-Thought Prompt
Vary your response depending on the situation:
Case 1: If the fever follows a patterns such as ~, suspect ~.
Case 2: If the user traveled to ~, identify ~.
Sub-Queries
How often are the symptoms?
Have you traveled recently?
Sub-Query Embeddings
Private Data
Has a fever every two days.
Recently traveled to Africa.
Untrusted Zone
LLM Server
Powerful Large Language Model
① Socratic CoT & Sub-Query Generation
DB Server
Alice's Private Database
③ Homomorphically Encrypted Vector Similarity Search
Bot: Considering the two-day symptom cycle and recent travel to Africa, malaria may be the cause.

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.

[*] Equal contribution (alphabetical order). Code is available at `https://github.com/Yubeen-Bae/PPMI`.

Preprint. Under review.

1 Introduction

Large language models (LLMs) are becoming the default backend for personal agents that manage emails, schedule meetings, and process real-time health data from wearable devices [63, 49, 72]. These agents must integrate data from heterogeneous sources—many stored remotely in cloud databases—using retrieval-augmented generation (RAG) [42]. While forwarding user queries along with retrieved data to powerful yet untrusted LLMs enhances performance, it introduces substantial privacy risks by potentially exposing private records [84, 36]. Conversely, restricting these operations to local trusted devices significantly degrades performance [51]. This raises the question: *Can we perform LLM interactions on private data while maintaining efficiency and accuracy without privacy risks or significant performance degradation?*

Existing privacy-preserving methods, such as data minimization or scrubbing personally identifiable information (PII), often sacrifice data utility or provide limited privacy through superficial suppressions [80]. To bridge the privacy-utility gap, we propose a four-stage hybrid framework that clearly delineates trusted and untrusted environments (left and right sides of Figure 1), ensuring private raw data either remains strictly within local boundaries or is securely encrypted when externally stored or searched. We integrate two novel components: (1) **Socratic Chain-of-Thought Reasoning**, which enables challenging yet non-private queries to be offloaded to a powerful external language model; and (2) a **Homomorphically Encrypted Vector Database**, a cryptographic system that allows efficient semantic search over encrypted records without ever decrypting them. This enables users to leverage cloud storage and compute resources while maintaining complete privacy—the cloud provider can execute searches without learning anything about the data content or search queries.

Stage 1: When the user, Alice, poses a query (see Figure 1, top), our Socratic Chain-of-Thought Reasoning elicits a detailed Chain-of-Thought prompt and sub-queries from a powerful external LLM. We provide only the main query, which we assume to be non-private in our protocol, to the external LLM without exposing any private user data. Rather than directly providing a diagnosis, we prompt the powerful LLM to generate Chain-of-Thought prompt for reasoning and targeted sub-queries for retrieval—in this case, questions about medications and travel history. This approach allows the powerful model to break down complex task into simpler ones, making it easier for the weaker local model to reason effectively when given access to private data as context. **Stage 2**: These sub-queries are then locally embedded to prepare them for secure semantic search over our encrypted vector database containing Alice's relevant records.

Stage 3: Once the sub-query embeddings reach our Homomorphically Encrypted Vector Database, the system executes secure vector similarity search, where all key vectors are homomorphically encrypted and compared against a million encrypted key vectors. Our novel inner product protocol computes similarity entirely in the encrypted domain in under one second using standard CPUs. The system then retrieves the corresponding encrypted records, returning top-k matches from a million-entry store in encrypted format. **Stage 4**: Finally, a much smaller, weaker language model operating exclusively within the local trusted zone generates the final response, drawing on both the chain-of-thought prompt and the decrypted private records supplied by the stronger remote model.

We extensively evaluate our framework on two long-context QA benchmarks. *LoCoMo* assesses recall of extensive conversational histories [53], while *MediQ* tests interactive clinical reasoning [46]. We establish two baselines representing privacy extremes: (1) a local-only (fully private) baseline using Llama-3 with 1B and 3B parameters, and (2) a remote-only (fully non-private) golden baseline using GPT-4o and Gemini-1.5-Pro. Our approach provides a balanced trade-off between these extremes.

Through our **Socratic Chain-of-Thought Reasoning**, the Llama 1B-parameter local model achieves an F1 score of 87.7 on LoCoMo, notably surpassing GPT-4o by 7.1 percentage points and the local-only baseline by 23.1 percentage points. This improvement likely stems from additional test-time computation enabled by the chain-of-thought process [16]. For MediQ, improvements are relatively smaller due to domain-specific adaptation challenges. Our **Homomorphically Encrypted Vector Database** efficiently searches entries from 10^6 records in under one second on commodity CPUs, maintaining $>99\%$ Recall@5 with a median storage overhead of just $\mathbf{5.8}\times$. *Collectively, our findings mark an important step toward privacy-preserving systems that effectively partition tasks between untrusted high-capacity LLMs and trusted lightweight local models, without requiring any additional post-training.*

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 [83, 63]. 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 [42].

Privacy Problem Formulation: While powerful cloud-based LLMs offer superior reasoning capabilities, they require users to expose private data to untrusted providers [57]. 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 [78].

Threat Model: We protect against three adversaries: (1) the LLM provider who receives user queries, (2) the database provider storing encrypted records [8], and (3) external attackers who may compromise these services [33]. Even with standard encryption, providers typically hold decryption keys, enabling potential privacy breaches through insider threats or security compromises [13, 31].

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 [25, 67].

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

We present our framework that enables powerful LLM reasoning while maintaining strict privacy guarantees, ensuring that sensitive user data is never exposed during interaction. This section describes our overall approach, which combines Socratic Chain-of-Thought Reasoning with our privacy-preserving framework designed to separate trusted and untrusted zones (Section 3.1). Section 4 further details our homomorphically encrypted retrieval system, which supports secure access to private records without compromising confidentiality.

3.1 Framework Overview

Figure 1 illustrates our framework's architecture, which separates computation into trusted and untrusted zones to balance privacy and performance. In the trusted zone (left side of the figure), the user's local device hosts a lightweight language model and embedding model with exclusive access to decryption keys, ensuring that sensitive data never leaves the user's control in plain form. The untrusted zone (right side of the figure) comprises cloud providers hosting: (1) a powerful LLM for abstract reasoning, and (2) an encrypted vector database storing the user's private records using homomorphic encryption [25, 12], allowing secure processing without data decryption.

Consider the medical consultation example in Figure 1: when a user asks "Why do I keep having fever?", the query flows to the remote LLM without exposing any private medical history. The powerful model generates targeted sub-queries (e.g., symptom frequency, travel history) that guide retrieval from the encrypted database, where personal records remain protected even during search operations thanks to homomorphic encryption. This architectural separation provides both active control—users explicitly manage what reaches remote models—and passive control, where cryptographic protection ensures data remains secure even if users make mistakes [27].

3.2 Framework Operation

The following detailed example illustrates our framework's operation, as shown in Figure 1:

1. The process begins when a user submits a query x:

   Why do I keep having fever?

2. Given the user's input x, the remote LLM generates:
   - A Chain-of-Thought (CoT) prompt c via its CoT generator G_c:

     Vary your response depending on the situation:
     Case 1: If the fever follows patterns such as ~, suspect ~.
     Case 2: If the user traveled to ~, identify ~.

   - Relevant sub-queries via its sub-query generator G_q:

     How often are the symptoms?
     Have you traveled recently?

3. The local client embeds these sub-queries and executes encrypted search on the user's private database $\mathcal{D}$, using a retriever R to obtain records v:

   Has a fever every two days.
   Recently traveled to Africa.

4. Finally, the local model L integrates the CoT prompt c and retrieved records v to generate the final response y:

   Considering the two-day symptom cycle and recent travel to Africa,
   malaria may be the cause.

We have provided examples of our prompts and the chains in Appendix D. To formalize this process, let $\mathcal{V}$ be the set of tokens and define k-tuples of $\mathcal{V}$ as:

$$\mathcal{V}^k = \{(v_0, \ldots, v_{k-1}) \mid v_0, \ldots, v_{k-1} \in \mathcal{V}\}$$

Then $\mathcal{V}^* = \bigcup_{k=0}^{\infty} \mathcal{V}^k$ is the set of all finite-length sequences.

We denote the Chain-of-Thought generator as G_c, the sub-query generator as G_q, and the retriever as R, which operates on a database $\mathcal{D}$. The local model L generates a response y based on an input x, using the CoT prompt and retrieved records as context:

$$y = L(x, c, v, h)$$

where $x, y, c, v, h \in \mathcal{V}^*$. Specifically:

- x denotes the user's input query
- $c = G_c(x)$ represents the CoT prompt generated by the remote model
- $v = R(G_q(x), \mathcal{D})$ indicates the retrieved records obtained by querying the encrypted database $\mathcal{D}$ with sub-queries generated by G_q
- h represents optional historical context (e.g., previous conversation turns), defaulting to an empty tuple $()$ if not provided
- y is the final response generated by the local model

This decomposition ensures that remote models G_c and G_q operate only on non-private data, while private records in $\mathcal{D}$ remain encrypted and are processed only within the trusted local environment by L. The next section details our homomorphically encrypted vector database that enables efficient retrieval without compromising privacy.

4 Homomorphically Encrypted Vector Database

In this section, we discuss the design and implementation of a vector database operating over encrypted data, integrating Homomorphic Encryption (HE) and Private Information Retrieval (PIR) techniques to enable secure and efficient semantic search with rapid updates. We begin by exploring the necessity for remote encrypted vector databases and their setup. Subsequently, we analyze existing HE-based inner product (IP) computations, proposing enhancements that significantly improve efficiency with faster updates. Finally, we present a detailed framework and its corresponding API specifications, illustrated clearly through algorithmic tables.

4.1 Motivations and Setup

The performance of personal assistants powered by language models significantly improves when relevant user-context data is appropriately provided. Thus, seamless integration and accumulation of user data are crucial for developing powerful personal assistants. While storing data locally on the user's device allows quick retrieval, local storage is inherently limited in capacity. Consequently, leveraging cloud-based solutions becomes essential, offering extensive storage capabilities and seamless data accumulation. Moreover, cloud solutions effectively handle multi-device scenarios by integrating data from diverse sources, such as wearable devices, and provide a unified environment, simplifying overall data management compared to fragmented local storage approaches.

Consider a simple yet inefficient baseline protocol for implementing a remote encrypted vector database: whenever a client needs to search or update an entry, it downloads the entire database from the server, decrypts it locally, performs the necessary operations, encrypts the entire database again, and uploads it back to the server. Although straightforward, this approach incurs significant communication overhead and computational burden on the client, rendering it impractical for large-scale applications.

To address these inefficiencies, we aim to design a remote vector database system that maintains the same robust security guarantees as the naive approach—where the database remains encrypted under a symmetric key held exclusively by the client—but achieves significantly better efficiency in communication and client-side computation.

The retrieval process within a vector database typically involves two critical sub-processes: search and return. The search phase computes similarity scores between a query vector and the key vectors stored in the database, and selects the top-k most relevant entries. In the return phase, corresponding data values are fetched from the database based on the selected entry identifiers (ids).

To ensure the robust security level of the baseline, three main processes must be executed in an oblivious manner: inner product (IP) computations, top-k selection, and data access. Homomorphic encryption (HE) is particularly effective for inner product calculations, as it significantly reduces both communication rounds and client-side computation. However, top-k selection, involving numerous logical comparisons, becomes computationally intensive when directly implemented with HE [30]. Therefore, we adopt a client-aided approach, enabling the client to efficiently select the top-k entry ids without excessive computational overhead.

Finally, once the client identifies the relevant entry ids, records are securely retrieved under the same security guarantees as the naive baseline. We use private information retrieval (PIR) protocols to fetch values corresponding to these ids without revealing which database entries are accessed.

To further enhance the efficiency of the database, particularly regarding frequent updates, we propose a novel HE-based IP algorithm that balances rapid updates and efficient search performance. Existing sublinear PIR schemes that rely on preprocessing are impractical for dynamic databases due to high preprocessing costs [43]. To mitigate this issue, there are several line of works [55, 58], single-server PIR protocol that operates efficiently without preprocessing, thereby enabling dynamic and rapid updates alongside secure and efficient searching.

4.2 Secure Inner Product, Technical Overview

Given a power-of-two integer $d > 1$, let $\mathcal{R}_{*,d} = \mathbb{Z}[X]/(X^d + 1)$. Given an integer $q > 0$, let $\mathcal{R}_{q,d} = \mathbb{Z}_q[X]/(X^d + 1) \simeq \mathcal{R}_{*,d}/q\mathcal{R}_{*,d}$. Polynomials are written in roman (e.g. q, k) and vectors are written in bold (e.g. $\mathbf{q}, \mathbf{k}$). Given a vector $\mathbf{v} \in F^t$, $v[i]$ denotes the i-th coordinate. Given polynomials $\mathrm{p}, \mathrm{p}' \in \mathcal{R}_{*,d}$, $\mathrm{p} \cdot \mathrm{p}' \in \mathcal{R}_{*,d}$ denotes the ring multiplication in $\mathcal{R}$. Given a polynomial $\mathrm{p} \in \mathcal{R}_{*,d}$, $\mathrm{p}[i]$ denotes the coefficient of X^i. Given a polynomial in $\mathrm{p} \in \mathcal{R}_{*,d'}$, we denote $\tilde{\mathrm{p}} \in \mathcal{R}_{*,d}$ as the natural embedding $\tilde{\mathrm{p}}(X) = \mathrm{p}(X^{d/d'})$. As we use d as the fixed RLWE dimension, we omit d in the notation $\tilde{\mathrm{p}}$.

The most significant difference between *semantic search* and a *vector database* is that the database must be dynamic, supporting insertion and deletion. An important observation is that HE operations should ideally not be used for insertion and deletion, as they accumulate errors and eventually corrupt the message.[2] Many existing HE-based inner-product algorithms are unsuitable for scenarios requiring dynamic updates. Current solutions for *encrypted semantic search* with a public database, such as Wally [3] and HERS [23], typically precompute key vectors in plaintext domain for fast search. However, this plaintext precomputation restricts dynamic updates in the ciphertext domain. In HERS, for instance, each key data point is distributed across different ciphertexts, necessitating complex homomorphic encryption (HE) operations for inserting or deleting keys along with their approximate values. This process can degrade data integrity over time due to accumulated errors resulting from frequent HE computations.

One way to avoid HE computations during insertion and deletion is to assign one ciphertext per key, allowing insertion and deletion by simply appending or removing ciphertexts. We designed a dedicated HE-IP scheme for this scenario, achieving both **exact updates** and **fast search**.[3]

The search process begins by computing the inner product between the query and the stored key vectors. Let us break down each step to derive the complete algorithm. For simplicity, we first solve the case where $n = d$ and r is a power of two. For $n \geq d$, we can extend the base case to compute multiple similarity scores. We describe the behavior of the underlying plaintexts.

Inner Product. Let the query vector be $\mathbf{q} = [\xi_i]_{0 \leq i < r} \in \mathbb{R}^r$ and the key vector be $\mathbf{k} = [\kappa_i]_{0 \leq i < r} \in \mathbb{R}^r$. The corresponding plaintext polynomials are encoded as

$$\mathrm{q}(X) = \sum_{i=0}^{r-1} q_i \cdot X^{-si} = \sum_{i=0}^{r-1} \lfloor \Delta \cdot \xi_i \rceil \cdot X^{-si} \in \mathcal{R}_{*,d}$$

and

$$\mathrm{k}(X) = \sum_{i=0}^{r-1} k_i \cdot X^{si} = \sum_{i=0}^{r-1} \lfloor \Delta \cdot \kappa_i \rceil \cdot X^{si} \in \mathcal{R}_{*,d},$$

where $\Delta > 0$ is a scaling factor and $d = rs$. Here the inner product $\langle \mathbf{q}, \mathbf{k} \rangle$ can be (approximately) derived as

$$\frac{1}{\Delta^2} \cdot (\mathrm{q} \cdot \mathrm{k})[0] \simeq \langle \mathbf{q}, \mathbf{k} \rangle.$$

We denote the (scaled) score σ as $\sigma = (\mathrm{q} \cdot \mathrm{k})[0]$. To pack multiple scores in a ciphertext for reducing communication, we extract the constant term from the ciphertext. We slightly modify the conventional homomorphic trace and write

$$\sum_{i=0}^{r-1} \varphi_i(\mathrm{q} \cdot \mathrm{k}) = r \cdot \sigma \tag{1}$$

where $\varphi_i = \mathrm{p}(X) \mapsto \mathrm{p}(X^{2i+1})$ is an automorphism over $\mathcal{R}_{*,d}$ for each $0 \leq i < r$.

Batching. We pack d scores $\sigma_0, \sigma_1, \ldots, \sigma_{d-1}$ into a single ciphertext. By Equation 1,

$$r \cdot \sum_{j=0}^{d-1} \sigma_j X^j = \sum_{j=0}^{d-1} \sum_{i=0}^{r-1} \varphi_i(\mathrm{q} \cdot \mathrm{k}_j) X^j = \sum_{i=0}^{r-1} \left[\varphi_i(\mathrm{q}) \cdot \left(\sum_{j=0}^{d-1} \varphi_i(\mathrm{k}_j) X^j \right) \right] \tag{2}$$

where $\sigma_j = (\mathrm{q} \cdot \mathrm{k}_j)[0]$ for each $0 \leq j < d$. Here we observe that the last term can be interpreted as an inner product between $(\varphi_i)_i$ and $\left(\sum_{j=0}^{d-1} \varphi_i(\mathrm{k}_j) X^j \right)_i$, separating query and key operations. The number of automorphisms for the query is independent of n (when $n \geq d$), and we can precompute (i.e., cache) the keys.

[2] One may consider using bootstrapping [24] to clean the errors, but it is almost infeasible due to its high computational cost.

[3] CHAM [64] also supports exact updates but is far less efficient than ours.

Caching. The key observation is that from Equation 2,

$$\sum_{j=0}^{d-1} \varphi_i(\mathrm{k}_j) X^j = \varphi_i\left(\sum_{j=0}^{d-1} \mathrm{k}_j X^{j\cdot\mathtt{inv}(i)}\right)$$

where $\mathtt{inv}(i) = (2i+1)^{-1} \bmod 2d$ so that $\varphi_i(X^{\mathtt{inv}(i)}) = X$.

This formula allows us to compute the automorphism φ_i only once. Therefore, we can significantly reduce the number of (homomorphic) automorphisms from $d\log(r)$ to $r-1$.

Butterfly Decomposition. For $\tilde{\mathbf{k}} = \left(\tilde{k}_j\right)_{0\le j<d} \in \mathcal{R}_{q,d}^d$ and $\mathbf{k} = \left(\sum_{j=0}^{d-1} \tilde{k}_j X^{j\cdot\mathtt{inv}(i)}\right)_{0\le i<r} \in \mathcal{R}_{q,d}^r$, let

$$\mathbf{M} = P \cdot \begin{bmatrix} X^0 & X^1 & X^2 & \cdots & X^{(d-1)} \\ X^0 & X^3 & X^6 & \cdots & X^{3(d-1)} \\ X^0 & X^5 & X^{10} & \cdots & X^{5(d-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X^0 & X^{2r-1} & X^{2(2r-1)} & \cdots & X^{(2r-1)(d-1)\}} \end{bmatrix} \in \mathcal{R}_{q,d}^{r\times d}$$

where $P \in \mathcal{R}_{q,d}^{r\times r}$ is a permutation matrix that corresponds to the permutation $i \mapsto \frac{(2i+1)^{-1}-1}{2} \bmod r : \{0,1,\ldots,r-1\} \to \{0,1,\ldots,r-1\}$. Then $\mathbf{k} = \mathbf{M}\tilde{\mathbf{k}}$ holds.

Multiplying $\mathbf{M}$ to $\tilde{\mathbf{k}}$ requires $r(r-1)$ polynomial additions, which is not negligible. Therefore, we use a DFT-style butterfly decomposition to reduce the computational cost.

Define $\mathbf{k}' \in \mathcal{R}_{q,d}^r$ as follows:

$$\mathbf{k}'[i] = \sum_{j=0}^{s-1} \tilde{k}_{j+si} X^j$$

for $0 \le i < r$. Then for $\mathbf{k}'' = \mathbf{B}\mathbf{k} \in \mathcal{R}_{q,d}^r$,

$$\varphi_{i,r}(\mathbf{k}''[i]) = \varphi_i(\mathbf{k}[i])$$

holds for $0 \le i < r$, where

$$\mathbf{B} = P \cdot \begin{bmatrix} X^0 & X^s & X^{2s} & \cdots & X^{(d-s)} \\ X^0 & X^{3s} & X^{6s} & \cdots & X^{3(d-s)} \\ X^0 & X^{5s} & X^{10s} & \cdots & X^{5(d-s)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X^0 & X^{s(2r-1)} & X^{2s(2r-1)} & \cdots & X^{(2r-1)(d-s)\}} \end{bmatrix} \in \mathcal{R}_{q,d}^{r\times r} \tag{3}$$

and $\varphi_{i,r} : \mathcal{R}_{q,d} \to \mathcal{R}_{q,d}$ is a permutation on the coefficients that satisfies

$$\varphi_{i,r}(\mathrm{p})[(2i+1)\cdot s\cdot u + j] = \mathrm{p}[s\cdot u + j]$$

for $0 \le j < s-1$ and $0 \le u < r$. By leveraging the butterfly matrix decomposition, we reduce the number of polynomial additions to $r\log(r)$. The detailed algorithm is written in Algorithm 7.

Removing the Leading Term r. To remove the leading term r from the result $r \cdot \sum_{j=0}^{d-1} \sigma_j X^j$, we multiply r^{-1} (mod q) before automorphisms.

$$r \cdot \sum_{i=0}^{r-1}\left(\varphi_i(r^{-1}\cdot \mathrm{q}) \cdot \left[r \cdot \varphi_i\left(\sum_{j=0}^{d-1} r^{-1}\cdot \mathrm{k}_j X^{j\cdot\mathtt{inv}(i)}\right)\right]\right) = r \cdot \sum_{j=0}^{d-1} \sigma_j X^j$$

Therefore, $\sum_{i=0}^{r-1}\left(\varphi_i(r^{-1}\cdot \mathrm{q}) \cdot \left[r \cdot \varphi_i\left(\sum_{j=0}^{d-1} r^{-1}\cdot \mathrm{k}_j X^{j\cdot\mathtt{inv}(i)}\right)\right]\right) = \sum_{j=0}^{d-1} \sigma_j X^j$.

Optimizations. To enhance the performance of homomorphically encrypted vector databases, we incorporate several advanced techniques to optimize computation, storage, and accuracy. One key optimization is caching via key-query decoupling, which allows keys to be precomputed and cached independently of queries. This significantly reduces query response time by accelerating inner product computation. We also apply hoisting [29, 10] to efficiently decompose queries, minimizing computational overhead. This technique, when combined with MLWE (Module Learning With Errors) [5] and seed-based ciphertext generation, enables compact storage and efficient updates. Storage and update efficiency is further improved through batch processing and MLWE-based seeding strategies [9], which reduce ciphertext size and update costs. Finally, we improve numerical precision by removing leading constant terms [14] in homomorphic computations, resulting in more accurate query results. See Appendix A for detailed descriptions, and Section 5.3 for results on latency, storage, and accuracy.

4.3 Database Operations

See Table 1 for our database's API. With these APIs (functions), we achieve efficient search and support dynamic updates with $O(1)$ complexity.

Algorithm 1 `Init`

```
Require: public parameters pp
A: $\mathtt{sk} \leftarrow \mathtt{GenSK}(\mathtt{pp})$
A: $\mathtt{pk} \leftarrow \mathtt{GenPK}(\mathtt{pp})$
A: Send $\mathtt{pk}$ to Bob
```

Algorithm 2 `Search`

```
Require: query $q$, database $\mathcal{D}$
A: $\mathbf{q} \leftarrow E(q)$
A: $\mathsf{q} \leftarrow \mathtt{EncryptHE}(\mathbf{q})$
A: Send $\mathsf{q}$ to Bob
B: $\mathsf{s} \leftarrow \mathtt{Score}(\mathsf{q}, \mathcal{D}_{\mathtt{cache}})$
B: Send $\mathsf{s}$ to Alice
A: $\mathbf{s} \leftarrow \mathtt{DecryptHE}(\mathsf{s})$
A: $\mathcal{I} \leftarrow \mathtt{TopK}(\mathbf{s})$
```

Algorithm 3 `Return`

```
Require: record ids $\mathcal{I}$
A&B: $\{v\} \leftarrow \mathtt{PIR}(\mathcal{D}_{\mathtt{value}}, \mathcal{I})$
A: $\{v\} \leftarrow \{\mathtt{DecryptAES}(v)\}$
```

Algorithm 4 `Insert`

```
Require: set of records $\{v\}$, database $\mathcal{D}$
A: $\mathbf{k} \leftarrow E(v)$
A: $\mathsf{k} \leftarrow \mathtt{EncryptHE}(\mathbf{k})$
A: $v \leftarrow \mathtt{EncryptAES}(v)$
A: Send $\{(\mathsf{k}, v)\}$ to Bob
B: $\mathcal{D}_{\mathtt{num}} \leftarrow \mathcal{D}_{\mathtt{num}} + \mathtt{len}(\{(\mathsf{k}, v)\})$
B: $\mathcal{D}_{\mathtt{key}} \leftarrow \mathtt{Append}(\mathcal{D}_{\mathtt{key}}, \{\mathsf{k}\})$
B: $\mathcal{D}_{\mathtt{value}} \leftarrow \mathtt{Append}(\mathcal{D}_{\mathtt{value}}, \{v\})$
B: $\mathcal{D}_{\mathtt{cache}} \leftarrow \mathtt{ReCache}(\mathcal{D}_{\mathtt{cache}}, \mathcal{D}_{\mathtt{key}}, \{\mathsf{k}\})$
```

Algorithm 5 `Delete`

```
Require: record ids $\mathcal{A}$, database $\mathcal{D}$
A: Send $\mathcal{A}$ to Bob
B: $\mathcal{D}_{\mathtt{num}} \leftarrow \mathcal{D}_{\mathtt{num}} - \mathtt{len}(\mathcal{A})$
B: $\mathcal{D}_{\mathtt{key}} \leftarrow \mathtt{Switch}(\mathcal{D}_{\mathtt{key}}, \mathcal{A})$
B: $\mathcal{D}_{\mathtt{value}} \leftarrow \mathtt{Switch}(\mathcal{D}_{\mathtt{value}}, \mathcal{A})$
B: $\mathcal{D}_{\mathtt{cache}} \leftarrow \mathtt{ReCache}(\mathcal{D}_{\mathtt{cache}}, \mathcal{D}_{\mathtt{key}}, \mathcal{A})$
```

Table 1: Set of algorithms for homomorphically encrypted vector database operations.

These operations include initialization, encrypted search and retrieval, as well as insertion and deletion. We denote the client as **Alice (A)** and the server as **Bob (B)**. Public parameters pp are shared between them. The function `GenSK` generates secret keys used in Homomorphic Encryption (HE), Advanced Encryption Standard (AES), and Private Information Retrieval (PIR), while `GenSwk` produces the corresponding public keys of HE and PIR including switching keys for homomorphic operations.

The finite-length sequences, such as textual queries q and records v , are embedded using an encoder E. The vector database $\mathcal{D}$ maintains the following attributes: `num` (number of entries), `key` (stored key vectors), `value` (encrypted records), and `cache` (cached key vectors for efficient search). Encryption and decryption are performed using `EncryptHE`, `DecryptHE`, `EncryptAES`, and `DecryptAES`.

The `Score` function computes similarity scores over encrypted vectors, and `TopK` selects the top-k most relevant entries using a heap-based algorithm with $O(n \log k)$ complexity. Retrieved values are fetched securely using PIR protocols. See Appendix A for more details.

To support dynamic updates, we include auxiliary operations such as `len` (entry count), `Append` (inserting new entries), `Switch` (deleting entries by overwriting them with the last entry), and `ReCache` (refreshing the cached key vectors). These operations are executed in a batched manner and achieve constant-time complexity.

Security Guarantees. Key vectors are encrypted using CKKS [17]. Values in the vector database are encrypted using non-deterministic AES-256 encryption. The combination of HE and AES provides robust security of our vector database. That is, our database provides 128-bit IND-CPA security [68, 11] and is quantum-resistant [8, 56].

5 Experiments

In this section, we empirically validate the effectiveness of our privacy-preserving framework. The experiments are organized into three parts. We first present the overall performance of the full framework. We then conduct ablations on Socratic Chain-of-Thought Reasoning, isolating the contributions of sub-query generation and chain-of-thought generation. Finally, we examine the accuracy of our encrypted database and evaluate its efficiency and scalability in terms of latency and storage cost.

5.1 Main Results

Experiments are conducted on two question-answering benchmarks: LoCoMo [53], designed to simulate personal assistant scenarios, and MediQ [45], aimed at simulating medical consultation scenarios. Both tasks require retrieving relevant user-specific data and performing complex reasoning to generate an accurate final answer. We use DRAGON [47] to obtain embedding vectors, facilitating the retrieval of proper records related to each query. See Appendix B for more details on the experimental setup. Consequently, experiments evaluate whether the model's final responses, derived from the user's original query and stored personal data, align closely with the desired answers.

Baseline	Model	LoCoMo	MediQ
Remote-Only Baseline (oracle)	R1 GPT-4o	80.6	**81.8**
	R2 Gemini-1.5-Pro	84.2	69.8
	R3 Claude-3.5-Sonnet	**89.8**	79.3
Local-Only Baseline	L1 Llama-3.2-1B	64.6	32.1
	L2 Llama-3.2-3B	68.7	43.2
	L3 Llama-3.1-8B	68.8	47.5
Hybrid Framework w/ Socratic CoT (ours)	L1 + R1	87.7	59.7
	L1 + R2	85.1	49.7
	L1 + R3	84.3	58.0
	L2 + R1	85.9	60.7
	L2 + R2	79.8	52.9
	L2 + R3	74.6	59.0
	L3 + R1	87.9	59.5
	L3 + R2	88.0	52.1
	L3 + R3	86.1	59.6

Table 2: Benchmark results on the **LoCoMo** and **MediQ** datasets. LoCoMo is evaluated by F1 score, while MediQ is evaluated by exact match. *Takeaway: Our privacy-preserving framework significantly outperforms local-only baselines and approaches the performance of oracle baselines without privacy constraints.*

Our framework improves local-only baselines by up to +27.6 percentage points. As shown in Table 2, our framework consistently outperforms the local-only baselines on both the LoCoMo and MediQ datasets. By delegating complex reasoning to powerful remote models, we observe substantial gains in performance. Specifically, we see improvements of up to 23.1 percentage points on LoCoMo and 27.6 on MediQ when comparing each local model with its corresponding privacy-preserving variants. On average, our approach improves F1 by +19.8 percentage points on LoCoMo and exact match by +19.0 percentage points on MediQ over the local-only counterparts. These gains are especially notable in challenging scenarios requiring domain expertise, such as medical consultations. Despite operating under strict privacy constraints, our framework approaches—and in some cases surpasses—the performance of oracle baselines that operate without privacy constraints. This demonstrates the effectiveness of our approach in balancing strong privacy with high utility.

5.2 Ablations on Socratic Chain-of-Thought Reasoning

To better understand the source of performance gains from Socratic Chain-of-Thought Reasoning, we conduct two ablation studies on the LoCoMo [53] and MediQ [45] datasets.

Reasoning augmentation leads to substantial performance gains. Table 3 compares remote-only and local-only baselines, with and without Socratic Chain-of-Thought Reasoning. On LoCoMo, all methods benefit from reasoning augmentation: explicitly prompting the model to reason through intermediate steps leads to clear performance gains. For example, the local-only baseline improves from 64.6 to 82.0, a gain of +17.4 percentage points, while the remote-only baseline improves from 80.6 to 92.6, a gain of +12.0 percentage points. These results suggest that reasoning augmentation through Socratic Chain-of-Thought Reasoning is key to performance gains on LoCoMo.

Method	Model	LoCoMo	MediQ
Remote-Only Baseline	R1	80.6	**81.8**
Remote-Only Baseline w/ Socratic CoT	R1 + R1	**92.6**	67.3
Local-Only Baseline	L1	64.6	32.1
Local-Only Baseline w/ Socratic CoT	L1 + L1	82.0	32.5
Hybrid Framwork w/ Socratic CoT (ours)	L1 + R1	87.7	59.7

Table 3: The first ablation study for Socratic Chain-of-Thought Reasoning on the **LoCoMo** and **MediQ** datasets. LocoMo is evaluated by F1 score, while MediQ is evaluated by exact match. R1 is GPT-4o, and L1 is Llama-3.2-1B. *Takeaway: Reasoning augmentation through Socratic Chain-of-Thought Reasoning is the primary driver of performance gains.*

Delegating both sub-queries and chain-of-thought generation to more powerful models is key. Table 4 highlights two key observations by isolating the contributions of sub-query generation and chain-of-thought generation.

First, delegating sub-query generation significantly improves retrieval quality. On LoCoMo, using a smaller model (Llama-3.2-1B) for sub-query generation limits retrieval performance (Recall@5 = 21.8). When this task is handled by a more capable model (GPT-4o), performance nearly doubles to 44.1. This indicates that sub-query generation often requires deeper understanding and reasoning, which smaller models struggle to achieve. Furthermore, using ground-truth retrieval results boosts performance even more, implying that better sub-query generation—closer to the ideal target—can further enhance final answer quality. On MediQ, the amount of private data per user is so limited that most of the relevant records are retrieved even without high-quality sub-queries, reducing the impact of sub-query generation on overall performance.

Second, delegating chain-of-thought generation improves final response quality. On LoCoMo, without any chain-of-thought (N/A), the F1 score is 77.8. Incorporating chain-of-thought reasoning from the smaller model raises it to 85.4, and using GPT-4o improves it further to 89.3. These results demonstrate that guiding generation with reasoning augmentation produced by stronger models plays a critical role in achieving high answer quality. Meanwhile, on MediQ, augmenting reasoning without rich domain knowledge from remote models yields only marginal improvements. In this case, the dominant factor is qualified reasoning criteria generated with rich domain knowledge, which powerful remote models provide far more effectively than smaller local models. We provide a more detailed analysis of the MediQ results in Appendix F.

CoT Sub-Query	R1	L1	N/A
GT	89.3	85.4	77.8
R1 (GPT-4o)	87.7	84.7	73.9
L1 (Llama-3.2-1B)	84.9	82.0	64.6

(a) LoCoMo

CoT Sub-Query	R1	L1	N/A
All	60.4	32.1	31.4
R1 (GPT-4o)	59.7	31.8	33.2
L1 (Llama-3.2-1B)	58.6	32.5	32.0

(b) MediQ

Table 4: The second ablation study for Socratic Chain-of-Thought Reasoning on the **LoCoMo** and **MediQ** datasets. LocoMo is evaluated by F1 score, while MediQ is evaluated by exact match. Each row corresponds to a different sub-query generation method: For LoCoMo, GT uses ground-truth private data without sub-query generation (Recall@5=**100.0**), R1 uses GPT-4o (Recall@5=**44.1**), and L1 uses Llama-3.2-1B (Recall@5=**21.8**). For Mediq, All setup uses the full user history as input since no retrieval annotation is available, while R1 and L1 follow the same retrieval configuration as in LoCoMo. Each column corresponds to a different chain-of-thought generation method, where N/A indicates that chain-of-thought reasoning is not used. L1 is used for final response generation across all settings. *Takeaway: Delegating both sub-query and chain-of-thought generation to more powerful models is crucial for optimal performance.*

These findings suggest that local-only baselines, even without disclosing queries, are sufficient as effective personal assistants for casual tasks like LoCoMo. In contrast, specialized domains such as MediQ necessitate leveraging the advanced expertise embedded within powerful remote models to deliver high-quality answers. *Therefore, collaborating with remote models becomes essential for users seeking more accurate responses in expert domains.*

5.3 Homomorphically Encrypted Vector Database

Encrypted search retains > 99% accuracy. We evaluate the search accuracy of our encrypted database using benchmarks from LoCoMo [53], Deep1B [4], and LAION [69], covering a range of vector dimensions and domains. Results in Table 5 show that the system maintains high search fidelity across both plaintext-to-ciphertext and ciphertext-to-ciphertext inner product computations. In particular, when the query is in plaintext—a setting aligned with our privacy-preserving framework involving non-private queries and private data—the encrypted database achieves accuracy comparable to its unencrypted counterpart, with both mean and maximum inner product errors remaining minimal. Metrics such as 1-Recall@1, 1-Recall@5, and MRR@10 confirm that the top-k results from the encrypted database closely mirror those of the plaintext system. These results demonstrate that encrypted search can be performed with negligible impact on accuracy.

Dataset	Max Error	Mean Error	Std Error	MRR@10	1-Recall@1	1-Recall@5
Plaintext Query						
LoCoMo	3.11e-3	2.97e-3	3.31e-9	99.99	99.97	100
Deep1B	5.29e-5	6.42e-6	7.00e-11	99.97	99.96	99.99
LAION	1.06e-4	9.83e-6	1.36e-10	99.86	99.79	99.95
Ciphertext Query						
LoCoMo	5.31e-2	2.32e-2	2.31e-1	93.31	89.20	98.89
Deep1B	1.39e-3	1.71e-4	4.61e-8	99.59	99.21	99.97
LAION	2.70e-3	3.44e-4	1.87e-7	99.85	99.78	99.95

Table 5: Search accuracy across **LoCoMo** [53], **Deep1B** [4], and **LAION** [69] datasets, evaluated under two settings: when the query is in plaintext (top) and when the query is encrypted (bottom), with encrypted keys in both cases. *Takeaway: Our encrypted database preserves high retrieval accuracy, achieving near-parity with the fully plaintext setting (both query and key).*

Encrypted search scales to 1M entries with < 1 second latency. Despite the typical computational overhead of homomorphic encryption, our system achieves practical latency for large-scale vector similarity search. Figure 2 presents results on the Deep1B [4] dataset, showing that by leveraging efficient SIMD-style operations and low-precision arithmetic in ciphertext space, the system achieves linear scalability across database sizes from 1,000 to 1 million entries. *Even at the million scale, end-to-end latency remains under one second, including encryption, computation, and communication—even under a slow network.* This performance makes the encrypted database viable for real-time applications. Network overhead has become the primary source of latency, reflecting that computation—particularly homomorphic encryption—no longer constitutes the principal bottleneck. This improvement is evident in our evaluation on the 100K subset of LAION dataset [69], where encrypted search completes in 62 ms on the fast network and 251 ms on the slow network, compared to 76 ms and 931 ms with Compass [87]—yielding 1.2× and 3.7× speed-ups, respectively.

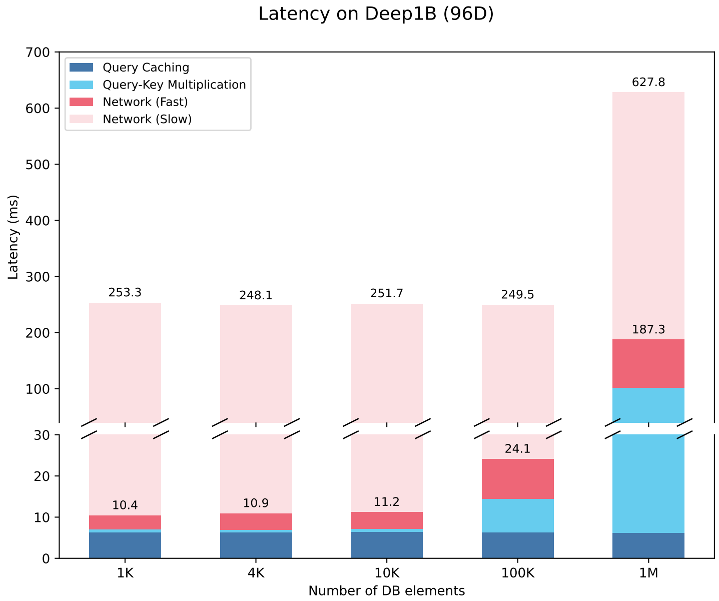

Figure 2: Multi-thread search latency (using 64 threads) breakdown on the Deep1B [4] dataset as the number of database entries increases. Red and pink bars represent network communication time on fast and slow networks, respectively, while the numbers above each bar indicate the corresponding latency. Blue bars represent query caching time; light-blue bars show query-key multiplication time. *Takeaway: Our encrypted search scales to 1M entries with < 1 second latency, as homomorphic operations incur relatively low overhead compared to network communication.*

Deep1B	**1K**	**4K**	**10K**	**100K**	**1M**
CHAM [64]	378 ms	389 ms	1,171 ms	9,406 ms	84,543 ms
Ours	150 ms	151 ms	156 ms	236 ms	951 ms

Table 6: Single-thread runtime of homomorphically encrypted matrix–vector multiplication as the number of vectors in the database increases. The CHAM [64] baseline is based on our re-implementation of the original method, incorporating additional optimizations such as ring packing and packing multiple vectors into a single ciphertext. *Takeaway: Our method achieves up to* 88× *speed-up over CHAM, enabling real-time encrypted search at million-scale.*

In addition, our system achieves significant improvements over previous homomorphic encryption methods. As shown in Table 6, our approach consistently delivers faster runtimes than CHAM [64], an encrypted matrix–vector multiplication method designed to support frequent updates. The performance gap widens with scale: while CHAM requires 84,543 ms to process 1 million entries, our method completes the same operation in just 2,280 ms—achieving a $\mathbf{37}\times$ speed-up. This efficiency primarily stems from our query caching strategy, which restructures the key-switching phase so that its computational complexity scales with the vector length rather than the full matrix size, effectively eliminating the dominant bottleneck in prior designs.

Encrypted storage incurs < $\mathbf{5.8}\times$ overhead. Storing high-dimensional vectors in homomorphic ciphertexts introduces nontrivial storage overhead. However, as detailed in Section 4 and Appendix A, our implementation adopts optimizations such as packing multiple vector components into a single ciphertext and omitting unused polynomial coefficients, effectively reducing space requirements. Moreover, we apply module-LWE variants and seed-based ciphertext generation techniques, which scale ciphertext size *linearly* with vector dimensionality rather than polynomial degree. As a result, the encrypted database achieves practical storage costs, less than $\mathbf{5.8}\times$ overhead even for millions of entries, enabling deployment in real-world systems without requiring excessive disk resources.

6 Related Work

In this paper, we address the challenge of privacy-preserving LLM interaction, focusing on protecting user records in the context, at inference time. Unlike private training approaches which safeguard the training corpus through techniques like DP-SGD and DP-ICL [1, 75], we focus on protecting user data provided as inputs to the model during inference, ensuring that sensitive context information remains confidential and is not leaked or memorized by the remote LLMs. Our work intersects with the following topics:

Private Inference via Encryption. Early approaches combined homomorphic encryption (HE) with neural networks, exemplified by CryptoNets [26], though with $10^3\times$ computational overhead. Subsequent systems like Gazelle [37] and XONN [66] reduced latency by hybridizing HE with garbled circuits and binary networks. Recent work extends these techniques to Transformers and LLMs: MPCFormer [44], PermLLM [85], and PUMA [20] achieve privacy for BERT and LLaMA architectures but still require seconds per token. Industry implementations like Apple's HE+PIR photo search [34] show promise, but cloud LLM providers have been reluctant to adopt these approaches due to significant computational overhead and complex key management.

Input Minimization and Sanitization Methods. Complementary approaches focus on sanitizing prompts before transmission. PREEMPT [18] detects and replaces sensitive spans with placeholders or differentially private values. PAPILLON [71] divides processing between local lightweight models and external LLMs, sending only abstracted prompts to the cloud. Additional work [21, 73] focuses on abstracting personal information. While effective for specific domains, these approaches typically require task-specific engineering or sacrifice accuracy when critical context is removed [80]. Our framework preserves task performance without cryptographic overhead by keeping raw data in the trusted zone while delegating only non-sensitive reasoning steps.

Chain-of-Thought Reasoning and Task Decomposition. Chain-of-thought (CoT) prompting has emerged as a powerful technique for improving LLM reasoning through step-by-step solutions. Zero-shot CoT techniques [79, 39] and task decomposition prompts [86, 62] guide models to break complex problems into manageable sub-problems, often enhanced with supervised reasoning traces [81]. Parallel work on model cascades aims to maximize efficiency by routing queries between different-sized models, as in FrugalGPT [15] and Hybrid LLM [19], typically using confidence estimators to determine when smaller models are insufficient [50, 28]. Multi-model frameworks like Socratic Models [82] and HuggingGPT [70] divide tasks between a powerful LLM planner and specialized executors, but assume the central model has full access to private data. In contrast, our approach performs test-time CoT decomposition without additional training while preserving privacy by ensuring the large LLM only sees abstracted queries rather than raw private data.

RAG and Agentic Workflows. Recent systems increasingly embed LLMs within persistent user-centric datastores to deliver personalized assistance. These range from research prototypes like Generative Agents [61] that maintain interaction histories as long-term memory, to commercial deployments such as ChatGPT's "Memory" and Operator [59, 60] that preserve multi-day conversation logs, and open frameworks like LangChain and LlamaIndex [54, 52] that provide memory backends as first-class primitives. Life-logging assistants like Rewind and Lindy [65, 48] index users' entire digital traces, leveraging the success of retrieval-augmented generation (RAG) [41] for grounding

LLMs in external knowledge. However, these systems typically assume trustworthy datastores, ignoring privacy risks highlighted by recent extraction and inference attacks [6]. Our framework is the first to combine an agentic RAG architecture with encrypted, local retrieval, addressing this critical privacy gap while maintaining the benefits of contextual personalization.

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.

Acknowledgments and Disclosure of Funding

This study was supported by the NAVER Digital Bio Innovation Research Fund, funded by NAVER Corporation (Grant No. 3720230180). Jaehyung Kim acknowledges support from the Stanford Graduate Fellowship, NSF, DARPA, and the Simons Foundation. Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 308–318, 2016. doi: 10.1145/2976749.2978318.

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

[3] Hilal Asi, Fabian Boemer, Nicholas Genise, Muhammad Haris Mughees, Tabitha Ogilvie, Rehan Rishi, Guy N Rothblum, Kunal Talwar, Karl Tarbe, Ruiyu Zhu, et al. Scalable private search with wally. *arXiv preprint arXiv:2406.06761*, 2024.

[4] 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.

[5] Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, Jai Hyun Park, and Damien Stehlé. Hermes: Efficient ring packing using mlwe ciphertexts and application to transciphering. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 37–69, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-38551-3.

[6] 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`.

[7] 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.

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

[9] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[10] 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.

[11] Jean-Philippe Bossuat, Rosario Cammarota, Ilaria Chillotti, Benjamin R. Curtis, Wei Dai, Huijing Gong, Erin Hales, Duhyeong Kim, Bryan Kumara, Changmin Lee, Xianhui Lu, Carsten Maple, Alberto Pedrouzo-Ulloa, Rachel Player, Yuriy Polyakov, Luis Antonio Ruiz Lopez, Yongsoo Song, and Donggeon Yhee. Security guidelines for implementing homomorphic encryption. Cryptology ePrint Archive, Paper 2024/463, 2024. URL `https://eprint.iacr.org/2024/463`.

[12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6(3):1–36, 2014.

[13] 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.

[14] 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.

[15] 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`.

[16] Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. In *International Conference on Learning Representations (ICLR)*, 2024.

[17] 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.

[18] 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.

[19] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*, 2024. URL `https://arxiv.org/abs/2404.14618`.

[20] 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.

[21] Yao Dou, Isadora Krsek, Tarek Naous, Anubha Kabra, Sauvik Das, Alan Ritter, and Wei Xu. Reducing privacy risks in online self-disclosures with language models. *arXiv preprint arXiv:2311.09538*, 2023.

[22] 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.

[23] Joshua J. Engelsma, Anil K. Jain, and Vishnu Naresh Boddeti. Hers: Homomorphically encrypted representation search. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 4(3):349–360, 2022. doi: 10.1109/TBIOM.2021.3139866.

[24] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585062. doi: 10.1145/1536414.1536440. URL `https://doi.org/10.1145/1536414.1536440`.

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

[26] 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.

[27] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2019.

[28] Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. *arXiv preprint arXiv:2404.10136*, 2024. URL `https://arxiv.org/abs/2404.10136`.

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

[30] Seungwan Hong, Seunghong Kim, Jiheon Choi, Younho Lee, and Jung Hee Cheon. Efficient sorting of homomorphic encrypted data with k-way sorting network. *IEEE Transactions on Information Forensics and Security*, 16:4389–4404, 2021. doi: 10.1109/TIFS.2021.3106167.

[31] 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.

[32] 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.

[33] 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.

[34] Apple Inc. Enhanced visual search with homomorphic encryption and PIR. Technical white-paper, October 2024. Retrieved January 2025 from `https://www.apple.com/legal/privacy/data/en/photos/`.

[35] 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.

[36] 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.

[37] Chiraag Juvekar, Vinod Vaikuntanathan, and Abhishek Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *USENIX Security Symposium*, pages 1651–1669, 2018.

[38] 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.

[39] 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`.

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

[41] 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. *arXiv preprint arXiv:2005.11401*, 2020. URL `https://arxiv.org/abs/2005.11401`.

[42] 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.

[43] Baiyu Li, Daniele Micciancio, Mariana Raykova, and Mark Schultz-Wu. Hintless single-server private information retrieval. Cryptology ePrint Archive, Paper 2023/1733, 2023.

[44] 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.

[45] 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.

[46] Shuyue Stella Li, Vidhisha Balachandran, Shangbin Feng, Jonathan S. Ilgen, Emma Pierson, Pang Wei Koh, and Yulia Tsvetkov. MediQ: Question-asking LLMs and a benchmark for reliable interactive clinical reasoning. *arXiv preprint arXiv:2406.00922*, 2024.

[47] 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.

[48] Lindy AI. Lindy — meet your ai assistant. `https://www.lindy.ai/lindy-agents/ai-assistant`, 2024. Accessed 16 May 2025.

[49] 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.

[50] Yueyue Liu, Hongyu Zhang, Yuantian Miao, Van-Hoang Le, and Zhiqiang Li. Optllm: Optimal assignment of queries to large language models. *arXiv preprint arXiv:2405.15130*, 2024. URL `https://arxiv.org/abs/2405.15130`.

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

[52] LlamaIndex Team. Llamaindex newsletter 2024-06-11: Enhanced memory modules boost agentic rag capabilities. `https://www.llamaindex.ai/blog/llamaindex-newsletter-2024-06-11`, June 2024. Accessed 16 May 2025.

[53] 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.

[54] Vasilios Mavroudis. Langchain. White paper, The Alan Turing Institute, 2024. URL `https://www.turing.ac.uk/sites/default/files/2024-11/langchain.pdf`.

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

[56] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

[57] 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.

[58] Muhammad Haris Mughees, Hao Chen, and Ling Ren. Onionpir: Response efficient single-server pir. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pages 2292–2306, 2021.

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

[60] OpenAI. Computer-using agent: Powering operator with a universal interface. `https://openai.com/index/computer-using-agent/`, January 2025. Accessed 16 May 2025.

[61] 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`.

[62] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022. URL `https://arxiv.org/abs/2210.03350`.

[63] 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.

[64] 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.

[65] Rewind AI. Rewind: Your ai assistant that has all the context. `https://www.rewind.ai/`, 2024. Accessed 16 May 2025.

[66] M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin E. Lauter, and Farinaz Koushanfar. XONN: Xnor-based oblivious deep neural network inference. In *28th USENIX Security Symposium*, pages 1501–1518, 2019.

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

[68] Phillip Rogaway. Nonce-based symmetric encryption. In *International workshop on fast software encryption*, pages 348–358. Springer, 2004.

[69] 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.

[70] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *arXiv preprint arXiv:2303.17580*, 2023. URL `https://arxiv.org/abs/2303.17580`.

[71] 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.

[72] 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.

[73] Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev. Large language models are advanced anonymizers. *arXiv preprint arXiv:2402.13846*, 2024.

[74] 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.

[75] Xinyu Tang, Richard Shin, Huseyin A Inan, Andre Manoel, Fatemehsadat Mireshghallah, Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, and Robert Sim. Privacy-preserving in-context learning with differentially private few-shot generation. *arXiv preprint arXiv:2309.11765*, 2023.

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

[77] 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.

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

[79] 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`.

[80] 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.

[81] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022. URL `https://arxiv.org/abs/2203.14465`.

[82] 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`.

[83] 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.

[84] 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.

[85] 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.

[86] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022. URL `https://arxiv.org/abs/2205.10625`.

[87] 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 6 and 7 describe the precomputations for the query and key, respectively, as mentioned right after Equation (2). Algorithm 8 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 as in Equation (2). 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) [40] 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 [14] that evaluates the trace without the leading term r, thereby improving the precision. This technique is applied for Line 2 of Algorithm 6 and Line 3 of Algorithm 7.
- **Hoisting** [29]: We adapt the hoisting technique that lazily computes the homomorphic operations to improve efficiency. Our adaptaion is similar to the double hoisting algorithm in [10]. Hoisting appears in the following instances.
  - Line 3 of Algorithm 6: For each index $0 \le i < s$, $\texttt{ModUp}(a_i)$ is computed only once.
  - Line 5,6 of Algorithm 6, Line 13,14 of Algorithm 7: We `ModDown` after summation, reducing the number of `modDown` to r per each j.
- **Reducing NTT dimension**: In Line 3,5,6 of Algorithm 6, 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 6 `Decompose`

```
Require: Query (seeded) MLWE ciphertext $(b, \rho)$ that encrypts $\mathrm{q} \in \mathcal{R}_{q,r}$ via the secret key $\mathbf{s} = (s_u)_{0 \le u < s} \in \mathcal{R}_{q,r}^s$. Here $b \in \mathcal{R}_{q,r}$ and $\rho$ 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 $\tilde{s_u}$ 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 $\rho$, 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: $\mathbf{a} = (a_u)_{0 \le u < s} \in \mathcal{R}_{q,r}^s \leftarrow \texttt{GenA}(\rho)$
2: $(b, \mathbf{a}) \leftarrow r^{-1} \cdot (b, \mathbf{a}) \bmod q$
3: $\hat{\mathbf{a}} = (\hat{a}_u)_{0 \le u < s} \in \mathcal{R}_{qp,r}^s \leftarrow (\texttt{ModUp}(a_u))_{0 \le u < s}$
4: for $j = 0$ to $r - 1$ do
5:     $ct_j \in \mathcal{R}_{qp,d}^2 \leftarrow \sum_{u=0}^{s-1} (\hat{a}_i \cdot \texttt{swk}_{j,u})$
6:     $ct_j \leftarrow \texttt{ModDown}(ct_j)$
7:     $ct_j \leftarrow \varphi_j(ct_j + (\tilde{b} \in \mathcal{R}_{q,d},\ 0))$
8: end for
9: return $(ct_j)_{0 \le j < r}$
```

Algorithm 7 `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 as defined in Equation 3.

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 8 `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 [55], 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 [7] and evaluated in in the same Google Cloud Platform configuration used by Compass [87] 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 [47] as the retriever because it outperforms other candidates, such as DPR [38], Contriever [35], and Instructor [74], 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) [32], Gemini-1.5-Pro (R2) [77], and Claude-3.5-Sonnet (R3) [2], 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) [22], 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 [53], 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 [45], 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 [4] and LAION [69], represented as 96-dimensional and 512-dimensional vectors respectively. For LoCoMo [53], 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 [76], 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 [7] and evaluated under the same Google Cloud Platform configuration used by Compass [87] 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 3. For response generation in the baselines, the prompt in Figure 4 was used. For Socratic Chain-of-Thought Reasoning, chain-of-thought generation was performed using the prompt in Figure 5, and response generation used the prompt in Figure 6. The prompts include substitution keys, which are described in Table 7.

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 7: 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 3: 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 4: 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 5: 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 6: Prompt used for response generation in the socratic chain-of-thought reasoning.

F Additional MediQ Analysis

As shown in Table 3, 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 7—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 7: Prompt used for chain-of-thought generation in the additional MediQ analysis.