Maturing Homomorphic Encryption to Enable Privacy Preserving Vector Search

Reza Rassool, Chair Kwaai

Kwaai Non-Profit Personal AI Lab

Overview

In today's data-driven society, the ability to share data securely is essential for scientific discovery and innovation. However, sharing sensitive information with collaborators and service providers introduces privacy risks, especially when the trustworthiness of those providers is uncertain. While significant research has advanced privacy-preserving data sharing algorithms (PPDSA), many solutions remain immature for practical deployment at scale. This project aims to bridge that gap by maturing and deploying homomorphic encryption and related techniques for privacy-preserving vector search, with a focus on Retrieval Augmented Generation (RAG) applications.

Motivation and Context

The need for scalable, privacy-preserving data sharing is widely recognized, as highlighted in recent literature and industry reports. Traditional approaches to data security focus on protecting data at rest and in transit, but data in use-when actively processed-remains vulnerable. Confidential computing and homomorphic encryption offer promising solutions by enabling computations on encrypted data, closing a critical gap in the data lifecycle[1][2][3].

Project Goals and Approach

Kwaai's project focuses on advancing privacy-preserving techniques for RAG systems. The central question is: **How can a data owner (Bob) send private data to a service operator (Eve) for vector database storage and querying, without exposing the data, while allowing users (Alice) to query it naturally?**

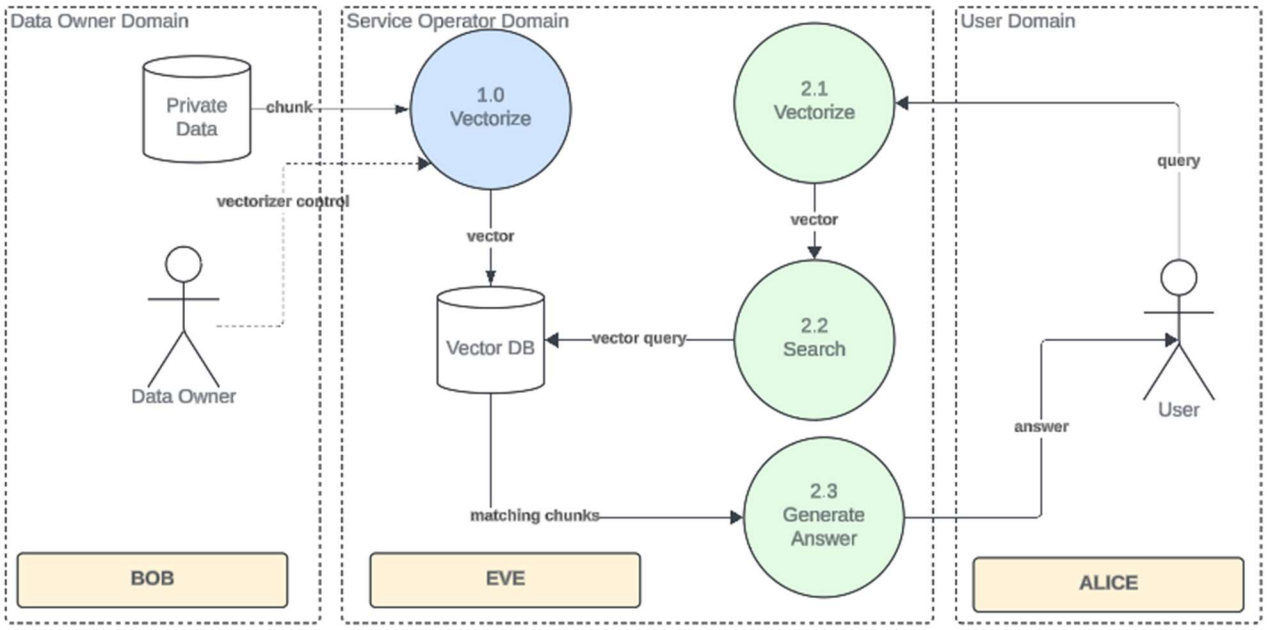


Figure 1 Typical RAG Flow Diagram

Security Roles and Trust Relationships

- **Data Owner (Bob):** Wants to share knowledge without exposing raw data.
- **Service Operator (Eve):** Provides compute infrastructure but may not be fully trusted.
- **User (Alice):** Queries the data and expects privacy for her queries.

This trust model reflects real-world scenarios where sensitive data must be processed by third parties, necessitating robust privacy guarantees[4][5][6].

Table 1	BOB	EVE	ALICE
BOB	Bob is the Data Owner who wants to offer a customer support, help desk, or Q&A bot informed by his knowledge base.	Bob does not completely trust Eve but uses her service of necessity because he lacks the necessary compute power. Bob is concerned that Eve may harvest his data to train her own service.	Bob has a defined relationship with Alice who may be his customer, patient or employee.
EVE	Eve would like to assure Bob that his data is safe with her and that she'll meet the SLA to keep Alice happy.	Eve is the Service Operator with the compute infrastructure to host vast databases and run a high-performance chatbot service.	Eve would like to host a fast and secure service so that Alice has a good customer experience.
ALICE	Alice values Bob's curated knowledge base and would like to query it in natural language.	Alice would prefer it if Eve did not eavesdrop and resell her queries to advertisers that would bombard her with ads.	Alice would like to query online knowledge bases of curated sources.

Technical Focus

Homomorphic Encryption (HE)

Homomorphic encryption allows computations to be performed directly on encrypted data, ensuring that results remain encrypted and can be decrypted only by authorized parties. This is particularly valuable for outsourced computation in untrusted environments, such as commercial cloud services[7][8][9][2].

- **Fully Homomorphic Encryption (FHE):** Supports arbitrary computations but is currently impractical for large-scale, real-time applications due to high computational overhead[7][10][11][2].
- **Partially/Additively Homomorphic Encryption (PHE/AHE):** Supports limited operations (e.g., addition, scalar multiplication) and is more efficient for vector similarity search, as demonstrated in recent research[7][10][8][11].

Recent studies show that AHE can efficiently support inner product similarity search-a core operation in vector retrieval-without requiring expensive ciphertext-ciphertext multiplications or bootstrapping, making it practical for real-world applications[7][8][11].

$f(x) = \xi'(f(\xi(x)))$, where
x is the data owned by Bob to be processed
f is the function to be outsourced for Eve to perform
ξ is the encryption operation that Bob performs on the data before submitting it to Eve
ξ' is the decryption operation that Bob performs on the result returned by Eve.

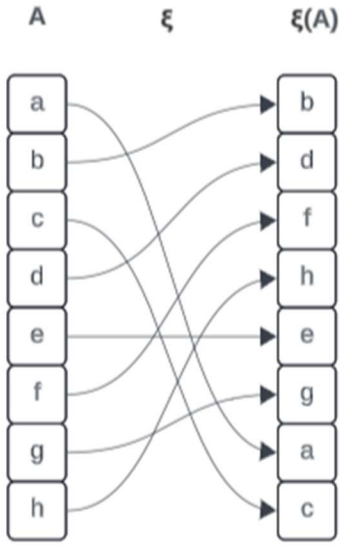
Securing Vector Search

Vector search is fundamental in RAG systems, powering applications like chatbots, recommendation engines, and federated learning. However, vector embeddings, while not directly revealing raw data, can be sensitive and vulnerable to reverse engineering[11][5].

Key Security Techniques:

Dimensional Scrambling: Permutes vector dimensions using a secret mapping, preserving mathematical operations while obscuring the underlying data. This technique leverages the commutative property of vector addition and is inspired by methods from telecommunications and signal processing[11].

Dimensional Scrambling



The vector $A = \{a, b, c, d, e, f, g, h\}$
The vector ξ represents a mapping of input dimensions to output dimensions. This is equivalent to a coordinate rotation. So $\xi(A) = \{b, d, f, h, e, g, a, c\}$

If vector $B = \{i, j, k, l, m, n, o, p\}$ then the dot product of the two vectors is
$A \cdot B = ai + bj + ck + dl + em + fn + go + hp$

Now, if both vectors are scrambled by ξ then
$\xi(A) \cdot \xi(B) = bj + dl + fn + hp + em + go + ai + ck$

So, it's clear that $\xi(A) \cdot \xi(B) = A \cdot B$
If we apply ξ to the vectors in the DB and to the vectorized query prompt then we can achieve homomorphically encrypted vector search where ξ is the secret key. For typical RAG the key would have 1536 factorial permutations.
The decryption is an identity matrix i.e. no operation needed.
Dimensional scrambling is well understood in the mobile industry as the basis for encoding signals. We will explore the obvious attack vectors and their remediation.

Noise Injection: Multiplies vectors by a secret noise vector, with a corresponding denoise operation applied to queries. This approach, rooted in classical signal processing, can further obscure data while maintaining search accuracy.

Clustering, GraphRAG, LoRA, and Sharding: These methods introduce additional secrecy by transforming, partitioning, or distributing data, with secret mappings known only to the data owner.

Recent advances, such as Cyborg's encrypted vector search engine and NVIDIA's cuVS library, demonstrate that end-to-end encrypted vector search is feasible with minimal performance overhead, making it suitable for regulated industries and IP-driven sectors[5].

Practical Deployment and Evaluation

The project emphasizes practical deployment over theoretical advances. Key evaluation criteria include:

- **Security:** Resistance to attack, key management, and compliance with frameworks like NIST and OWASP[1][4][9].
- **Practicality:** Performance (latency, throughput), ease of integration, and real-time responsiveness for chatbot applications[7][10][11].

- **Scalability and Sustainability:** Ability to scale with data size, minimize resource consumption, and reduce carbon footprint[12].

Kwaai's methodology includes agile development, continuous security audits, and open-source releases. The project leverages open-source libraries (e.g., LightPHE), vector databases (ChromaDB, Pinecone), and a large volunteer base for rapid prototyping and deployment.

Broader Impacts

Confidential Computing and Distributed AI

Confidential computing, using hardware-based Trusted Execution Environments (TEEs), is emerging as a complementary technology, enabling secure computation on sensitive data even in untrusted environments[1][4][6][3]. TEEs protect data in use by isolating code and data from the rest of the system, ensuring confidentiality even if the infrastructure is compromised.

Personal AI and Data Sovereignty

Kwaai's mission aligns with the movement toward Personal AI: empowering individuals to own and control their data and AI models, running locally or in trusted environments. This approach counters the trend of centralized, cloud-based AI services that require users to surrender privacy[12].

Research and Industry Leadership

Kwaai's team brings deep expertise from industry and open-source communities, focusing on pragmatic, deployable solutions. The project aims to bridge the gap between academic theory and industry needs, with an emphasis on ecological sustainability and democratization of AI.

Conclusion

Kwaai's project represents a significant advance in privacy-preserving AI, focusing on practical deployment of homomorphic encryption and related techniques for secure vector search in RAG systems. By integrating AHE, dimensional scrambling, noise injection, and confidential computing, the project aims to deliver scalable, secure, and sustainable solutions for real-world data sharing and AI applications. This work supports the broader vision of democratizing AI, protecting data sovereignty, and enabling secure, distributed intelligence for all[7][8][1][5][12][11][6][2][3].

References:

-[7] arXiv: A Note on Efficient Privacy-Preserving Similarity Search for Encrypted Vectors

-[10] arXiv: Encrypted Vector Similarity Computations Using Partially Homomorphic Encryption

-[8] arXiv: A Note on Efficient Privacy-Preserving Similarity Search for Encrypted Vectors

-[1] Google Cloud: Confidential computing for data analytics, AI, and federated learning

-[4] arXiv: C-FedRAG: A Confidential Federated Retrieval-Augmented Generation System

-[5] NVIDIA: Bringing Confidentiality to Vector Search with Cyborg and NVIDIA cuVS

-[9] Red Hat: Preserving privacy in the cloud: speeding up homomorphic encryption with FPGAs

-[12] Kwaai: About - Kwaai

-[11] Sefik Serengil: Vector Similarity Search with Partially Homomorphic Encryption in Python

-[6] arXiv: Privacy-Preserving Decentralized AI with Confidential Computing

-[2] AHIMA Journal: Moving Beyond Traditional Data Protection: Homomorphic Encryption

-[3] TechUK: Power of Confidential Computing - Fortifying Generative AI Adoption

⁂

1. https://cloud.google.com/architecture/confidential-computing-analytics-ai
2. https://journal.ahima.org/page/moving-beyond-traditional-data-protection-homomorphic-encryption-could-provide-what-is-needed-for-artificial-intelligence
3. https://www.techuk.org/resource/power-of-confidential-computing-fortifying-generative-ai-adoption.html
4. https://arxiv.org/abs/2412.13163
5. https://developer.nvidia.com/blog/bringing-confidentiality-to-vector-search-with-cyborg-and-nvidia-cuvs/
6. https://arxiv.org/html/2410.13752v1
7. https://arxiv.org/html/2502.14291v1
8. https://arxiv.org/abs/2502.14291
9. https://research.redhat.com/blog/article/privacy-in-the-cloud-speeding-up-homomorphic-encryption-with-fpgas/
10. https://arxiv.org/abs/2503.05850

11. https://sefiks.com/2025/03/04/vector-similarity-search-with-partially-homomorphic-encryption-in-python/

12. https://www.kwaai.ai/about

13. Refined with perplexity