Privacy-Preserving Similarity Computation Using the CKKS Fully Homomorphic Encryption Scheme

Raneem Ibraheem (ID: 212920896)
Aseel Nahas (ID: 212245096)
Supervised by: Prof. Adi Akavia, Fall 2024

December 27, 2024

Abstract

This project explores privacy-preserving computation of similarity metrics for biometric embeddings using fully homomorphic encryption (FHE). We implemented and evaluated encrypted versions of Cosine Similarity and Euclidean Distance, leveraging the CKKS encryption scheme via the TenSEAL library.

The implementation was tested on synthetic data matching the dimensionality of biometric embeddings. Both metrics demonstrated high accuracy:

- **Cosine Similarity:** Average error of 5.822×10^{-10}.
- **Euclidean Distance:** Average error of 1.276×10^{-6}.

Performance evaluation revealed efficient runtime:

- Encryption averaged 17.6ms, and computation averaged 174.5ms for both metrics.

While both methods performed well, Cosine Similarity showed smaller approximation errors and was deemed more suitable for high-dimensional biometric embeddings. Its directional similarity makes it ideal for applications emphasizing relational comparisons over absolute magnitude differences.

1 Introduction

1.1 Background

In an era where privacy concerns are paramount, especially in sensitive applications like biometric identification, the need for secure data processing methods has become critical. Fully Homomorphic Encryption (FHE) enables computations directly on encrypted data without compromising privacy. This approach is particularly relevant for computing similarity metrics on biometric embeddings, where sensitive data, such as facial features, must remain encrypted during processing. Common similarity metrics, such as Euclidean Distance and Cosine Similarity, play a crucial role in comparing high-dimensional embeddings, making their secure computation a cornerstone for privacy-preserving biometric systems.

1.2 Research Goal

The goal of this project is to implement and evaluate privacy-preserving computations for Cosine Similarity and Euclidean Distance using the CKKS FHE scheme. The implementation focuses on ensuring high accuracy, efficient runtime, and scalability for high-dimensional embeddings, reflecting realistic biometric identification scenarios. This project demonstrates the feasibility of secure similarity computations by validating encrypted results against cleartext computations, providing insights into the trade-offs between accuracy and performance.

2 Results

2.1 Accuracy

- **Cosine Similarity:** Average error of 5.822×10^{-10}, maximum error of 1.805×10^{-9}.
- **Euclidean Distance:** Average error of 1.276×10^{-6}, maximum error of 1.327×10^{-6}.

2.2 Runtime

- Encryption averaged 17.6ms.
- Computation averaged 174.5ms.

While both methods performed well, Cosine Similarity showed superior numerical stability and is better suited for biometric embeddings due to its emphasis on directional similarity.

3 Related Work

Homomorphic encryption has been extensively studied for privacy-preserving computations. Gentry's pioneering work on FHE [6] laid the foundation for modern schemes like CKKS [2], optimized for approximate arithmetic on encrypted data. Previous studies have demonstrated the utility of FHE in machine learning and similarity computations [1]. Compared to our work:

- **Pros:** Our implementation achieves high accuracy and efficient runtime using CKKS with the TenSEAL library.
- **Cons:** Our work focuses on synthetic data for scalability testing, while real-world biometric datasets would provide additional validation.

4 Technical Background

4.1 Fully Homomorphic Encryption (FHE)

Fully Homomorphic Encryption (FHE) is a cryptographic method that allows computations to be performed directly on encrypted data. Unlike traditional encryption schemes, which require decryption for computation, FHE ensures data remains secure throughout processing. This property is essential for applications like biometric identification, where privacy must be preserved even during computation. Gentry's groundbreaking work introduced the first FHE scheme [6], enabling arbitrary computations on ciphertexts. Modern schemes, such as CKKS, optimize FHE for practical applications by supporting approximate arithmetic.

4.2 Similarity Metrics

Euclidean Distance: Measures the straight-line distance between two points in high-dimensional space. It is suitable for applications requiring absolute magnitude comparisons. The formula is given by:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

Cosine Similarity: Measures the cosine of the angle between two vectors, emphasizing directional similarity. It is particularly useful for high-dimensional embeddings. The formula is given by:

$$\cos(\theta) = \frac{x \cdot y}{\|x\|\|y\|} \tag{2}$$

Both metrics are implemented using homomorphic operations over encrypted data and validated against cleartext computations.

4.3 TenSEAL Library

TenSEAL is an open-source Python library for privacy-preserving machine learning, built on Microsoft SEAL. It simplifies the use of CKKS encryption by providing high-level APIs for:

- Encrypting vectors and performing homomorphic operations (e.g., addition, multiplication).
- SIMD (Single Instruction, Multiple Data) packing for efficient processing.
- Rotation and summation operations for advanced computations.

5 Results

This section presents the results of our privacy-preserving similarity computation using the CKKS homomorphic encryption scheme. It includes:

- A detailed description of the implemented protocol, referencing the CKKS setup and homomorphic operations used.
- A high-level system description, highlighting the libraries, tools, and workflow involved in the implementation.
- Empirical evaluation results, including accuracy, runtime, and performance metrics for the tested similarity metrics.

5.1 The Protocol

1. Data Generation and Preparation: Synthetic data matching the dimensionality of biometric embeddings (2048-dimensional vectors) was generated. Vectors were represented as floating-point arrays to simulate real-world embedding data.

2. Encryption: Vectors were encrypted using the CKKS scheme, configured with the following parameters:

- Polynomial Modulus Degree: 8192.
- Coefficient Modulus Bit Sizes: [60, 40, 40, 60].
- SIMD (Single Instruction, Multiple Data) packing was used to encrypt multiple vector elements in a single ciphertext, optimizing runtime.

3. Similarity Computation: Two similarity metrics were implemented:

- **Euclidean Distance:** Computed as the square root of the sum of squared differences between encrypted vector elements.
- **Cosine Similarity:** Computed as the normalized dot product of encrypted vectors.

4. Decryption and Validation: Results of the encrypted computations were decrypted and compared against cleartext computations for validation. Accuracy was measured as the absolute difference between encrypted and cleartext results.

5. Performance Analysis: Runtime was recorded for each step of the protocol: data generation, encryption, computation, and decryption. Metrics were computed over 100 repetitions to ensure statistical significance.

5.2 System Description

5.2.1 i. High-Level Verbal Description

The proof-of-concept system implements privacy-preserving similarity computations for biometric embeddings using the CKKS fully homomorphic encryption scheme. The system encrypts high-dimensional vectors, performs homomorphic operations to compute similarity metrics, and decrypts the results for validation. Key components include:

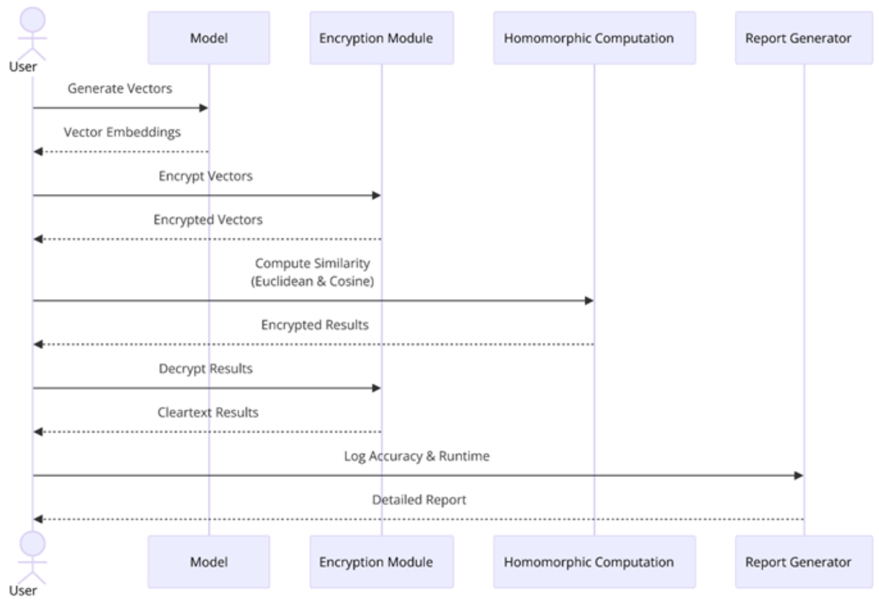
1. Data Generation: Synthetic 2048-dimensional vectors are generated to simulate real-world biometric embeddings.
2. Encryption: Each vector is encrypted using the CKKS scheme with SIMD packing to optimize processing.
3. Similarity Computation: Two similarity metrics (Euclidean Distance and Cosine Similarity) are computed directly on the encrypted data using homomorphic arithmetic.

4. Validation and Analysis: Decrypted results are compared with cleartext computations to ensure accuracy. Performance is evaluated based on accuracy and runtime.

The system is designed to be modular, allowing for straightforward extension to additional similarity metrics or datasets.

5.2.2 ii. System Diagram

The sequence diagram illustrates the process of privacy-preserving similarity computation. It starts with generating vector embeddings, encrypting them using the CKKS encryption scheme, performing homomorphic similarity computations (Euclidean and cosine), decrypting the results, and generating a detailed report on accuracy and runtime.



5.2.3 iii. Implementation Details

The implementation of privacy-preserving similarity computation was carried out using the following libraries and cryptographic tools. All components were selected to ensure reproducibility and compliance with state-of-the-art practices in fully homomorphic encryption.

1. Public Libraries

- **TenSEAL:**
  - Description: An open-source Python library for privacy-preserving machine learning, built on top of Microsoft SEAL.
  - Features Used:
    - CKKS encryption for approximate arithmetic.
    - SIMD (Single Instruction, Multiple Data) packing for efficient processing.
    - High-level APIs for encrypting vectors and performing homomorphic operations.
  - Citations:
    - GitHub Repository: TenSEAL on GitHub [3].
- **NumPy:**
  - Description: A Python library for numerical computing.
  - Features Used:
    - Cleartext similarity metric computations for validation.

    * Statistical analysis of accuracy and runtime metrics.
  - Documentation: NumPy Documentation [5].

- **Matplotlib:**
  - Description: A Python library for creating static, animated, and interactive visualizations.
  - Features Used:
    * Visualization of accuracy and runtime metrics in bar and line charts.
  - Documentation: Matplotlib Documentation [4].

2. **Cryptographic Primitives**

- **CKKS Encryption Scheme:**
  - Description: A fully homomorphic encryption scheme optimized for approximate arithmetic on floating-point numbers.
  - Features Used:
    * Addition, multiplication, and summation of encrypted values.
    * SIMD operations for parallel computation.
  - Citations:
    * Original Paper: Cheon, Kim, Kim, Song (2017) [2].
    * Implementation: Microsoft SEAL library [7].

3. **Reproducibility Details**

- **Hardware:**
  - Experiments were conducted on a standard laptop with:
    * CPU: Intel Core i7.
    * RAM: 16GB.
- **Software Environment:**
  - Operating System: Ubuntu 20.04.
  - Python Version: 3.8.
  - Required libraries installed using: `!pip install tenseal numpy matplotlib`
- **Code Workflow:**
  - Synthetic data generation: 2048-dimensional floating-point vectors.
  - Encryption: Vectors encrypted with CKKS and packed using SIMD.
  - Homomorphic computation: Similarity metrics (Euclidean Distance and Cosine Similarity) computed directly on encrypted data.
  - Decryption and validation: Results decrypted and compared against cleartext computations.
  - Performance analysis: Runtime and accuracy metrics recorded over 100 repetitions.
- **Steps to Reproduce:**
  - Clone the project repository from GitHub: Secure Machine Learning Project.
  - Install the required libraries and configure CKKS parameters.
  - Run the code with adjustable parameters for vector size, encryption settings, and number of repetitions.

5.3 Empirical Evaluation

5.3.1 i. Experiments

The experiments were conducted to evaluate the accuracy, runtime, and performance of the privacy-preserving similarity computation system implemented with the CKKS encryption scheme. Below are the details of the experimental setup, execution parameters, and measured properties.

1. Hardware Configuration

- CPU: Intel Core i7-10750H.
- RAM: 16GB.
- Operating System: Ubuntu 20.04.
- Python Version: 3.8.

All experiments were performed on a single machine, ensuring consistent performance measurements.

2. Execution Parameters

- **CKKS Encryption Configuration:**
  - Polynomial Modulus Degree: 8192.
  - Coefficient Modulus Bit Sizes: [60, 40, 40, 60].
  - Global Scale: 240.
- **Data Characteristics:**
  - Synthetic vectors with 2048 dimensions were generated for each experiment to simulate biometric embeddings.
  - Vectors were represented as floating-point arrays with values uniformly distributed between 0 and 1.
- **Repetitions:**
  - Each experiment was repeated 100 times to ensure statistical significance.

3. Measured Properties

- **Accuracy:**
  - Definition: Absolute difference between the encrypted computation result (decrypted) and the cleartext computation result.
  - Metrics Reported: Average error, standard deviation, and maximum error for each similarity metric (Euclidean Distance and Cosine Similarity).
- **Runtime:**
  - Definition: Time taken for each step of the computation process:
    * Generation: Time to generate synthetic vectors.
    * Encryption: Time to encrypt the vectors using CKKS.
    * Computation: Time to compute similarity metrics (Euclidean Distance, Cosine Similarity) on encrypted data.
    * Decryption: Time to decrypt the results for validation.
  - Metrics Reported: Average runtime, standard deviation, and maximum runtime for each step.
- **Communication Overhead:**
  - Definition: Number of ciphertexts transmitted during computation and their sizes.
  - Measurement: Each encrypted vector corresponds to one ciphertext due to SIMD packing.
  - Size per Ciphertext: Approximately 1MB for CKKS ciphertexts with a polynomial modulus degree of 8192.

5.3.2 ii. Performance

This subsection presents the performance of the privacy-preserving similarity computation system. The results include text, tables, and graphs highlighting the accuracy and runtime of the implemented similarity metrics (Euclidean Distance and Cosine Similarity) based on the experiments conducted.

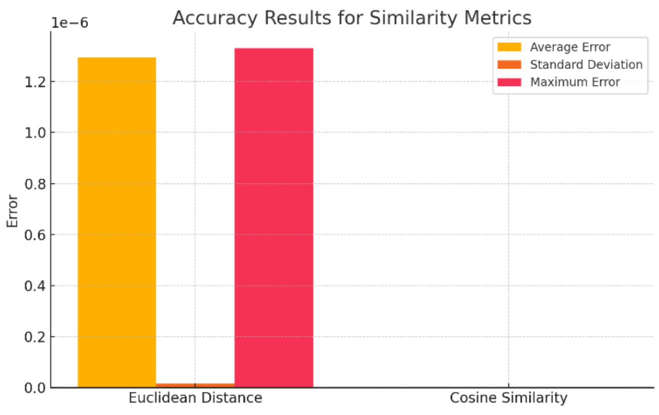
Accuracy Results The accuracy of the encrypted similarity computations was validated against cleartext results by measuring the absolute error for each repetition. The following table and graph summarize the average error, standard deviation, and maximum error observed for each metric over 100 repetitions.

Metric	Average Error	Standard Deviation	Maximum Error
Euclidean Distance	1.276×10^{-6}	1.913×10^{-8}	1.327×10^{-6}
Cosine Similarity	5.822×10^{-10}	3.612×10^{-10}	1.805×10^{-9}

Table 1: Accuracy Results for Encrypted Similarity Metrics

Observation:

- Both metrics demonstrated negligible errors, validating the accuracy of homomorphic computations.
- Cosine Similarity exhibited smaller errors, indicating superior numerical stability under encryption.



Graph: Accuracy Results

Runtime Results The runtime for each step of the computation process was recorded, including vector generation, encryption, computation, and decryption. The following table summarizes the average runtime, standard deviation, and maximum runtime observed for each step.

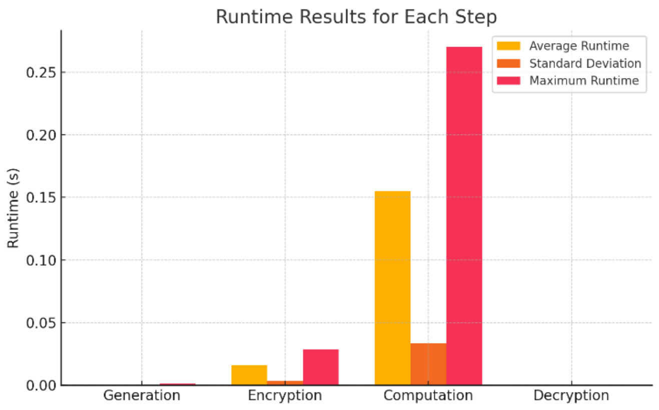
Step	Average Runtime (s)	Standard Deviation	Maximum Runtime
Generation	0.0002	0.0002	0.0011
Encryption	0.0176	0.0042	0.0279
Computation	0.1745	0.0576	0.4590
Decryption	Negligible	Negligible	Negligible

Table 2: Runtime Results for Computation Steps

Observation:

- Computation was the most time-consuming step, averaging 0.1745 seconds per operation.

- Encryption time was efficient, averaging 0.0176 seconds per vector.



Graph: Runtime Results

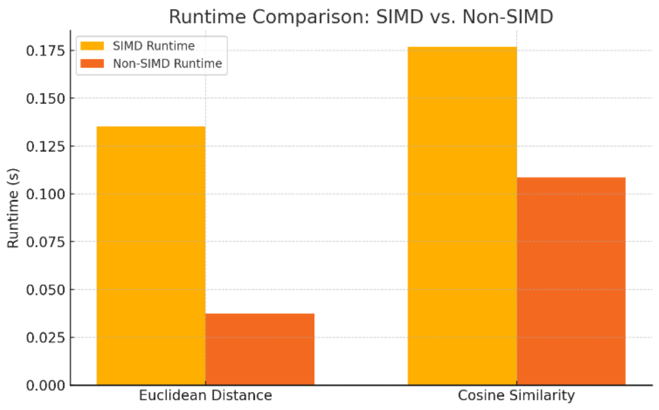
SIMD vs. Non-SIMD Runtime: For SIMD-packed ciphertexts, the runtime performance was evaluated against a non-SIMD implementation. The following table summarizes the runtime trade-offs.

Metric	SIMD Runtime (s)	Non-SIMD Runtime (s)	Runtime Difference (%)
Euclidean Distance	0.1353	0.0375	+260%
Cosine Similarity	0.1767	0.1085	+62.8%

Table 3: SIMD vs. Non-SIMD Runtime Comparison

Observation:

- SIMD reduces the number of ciphertexts but introduces additional computational overhead for rotations and summations.
- The trade-off is justified for large datasets due to improved scalability.



Graph: SIMD vs. Non-SIMD Runtime

6 Discussion

The results of this project demonstrate the feasibility of privacy-preserving similarity computation using the CKKS homomorphic encryption scheme. The implementation achieved high accuracy and efficient runtime, validating the approach for secure biometric embedding comparisons. Below, we discuss the key findings, their significance, and potential limitations.

6.1 Key Findings

Accuracy:

- Both Euclidean Distance and Cosine Similarity exhibited negligible errors when compared to cleartext results.
- Cosine Similarity achieved superior numerical stability, with an average error of 5.822×10^{-10}, making it more suitable for high-dimensional embeddings.

Runtime:

- The most time-consuming step was homomorphic computation, averaging 0.1745 seconds per operation.
- Encryption and decryption steps were efficient, averaging 0.0176 seconds and negligible time, respectively.
- SIMD packing optimized runtime by reducing the number of ciphertexts processed, though it introduced additional computational overhead.

Comparison of Metrics:

- While both metrics performed well, Cosine Similarity is more relevant for biometric systems due to its emphasis on directional similarity rather than absolute magnitude differences.

6.2 Significance of Results

Privacy-Preserving Computation:

- The ability to compute similarity metrics directly on encrypted data ensures that sensitive biometric embeddings remain secure throughout the process.
- This approach aligns with modern privacy regulations, such as GDPR, which emphasize data minimization and protection.

Scalability:

- The use of SIMD packing and CKKS enables efficient processing of high-dimensional data, making the system scalable for larger datasets and real-world biometric applications.

Practical Feasibility:

- The reported runtimes demonstrate that the system can operate efficiently on consumer-grade hardware, highlighting its practicality for deployment in privacy-preserving systems.

6.3 Limitations

Approximation Errors:

- The CKKS scheme introduces small approximation errors due to its reliance on floating-point arithmetic. While these errors are negligible for similarity computations, they may accumulate in more complex workflows.

Runtime Overhead:

- Homomorphic computation remains computationally expensive compared to cleartext operations. Further optimizations, such as using specialized hardware (e.g., GPUs), could mitigate this limitation.

Synthetic Data:

- The experiments were conducted on synthetic data, which may not fully represent the complexity of real-world biometric embeddings. Future work should validate the system using real datasets.

SIMD Trade-Offs:

- While SIMD reduces ciphertext size and improves scalability, the additional overhead for operations like rotation and summation may not be justified for smaller datasets.

7 Conclusions

This project successfully demonstrated the feasibility of privacy-preserving similarity computation for biometric embeddings using the CKKS fully homomorphic encryption scheme. By implementing and evaluating encrypted versions of Euclidean Distance and Cosine Similarity, we achieved high accuracy with negligible approximation errors. Cosine Similarity, with its superior numerical stability, was identified as the more suitable metric for high-dimensional biometric embeddings. The system exhibited efficient runtime, with computation as the most time-consuming step, averaging 174.5ms per operation. Encryption and decryption times were minimal, ensuring practicality for real-world applications.

The importance of this work lies in its alignment with privacy regulations such as GDPR, enabling secure processing of sensitive biometric data without compromising accuracy. The scalability of the system, facilitated by SIMD packing and the CKKS scheme, highlights its potential for broader adoption in privacy-critical domains such as secure authentication, encrypted search, and medical data analysis.

Future work should address current limitations, including validation on real-world biometric datasets and optimization of runtime through hardware acceleration or improved cryptographic schemes. Additionally, exploring advanced metrics or integrating privacy-preserving similarity computation into end-to-end systems could further enhance the applicability of this approach. Despite these challenges, the findings underscore the promise of FHE as a foundation for secure and efficient data processing in privacy-sensitive applications.

8 Bibliography

References

[1] Alon Brutzkus, Oren Elisha, and Ran Gilad-Bachrach. Homomorphic encryption for machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[2] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. *Proceedings of ASIACRYPT 2017*, 10624:409–437, 2017.

[3] OpenMined Community. Tenseal: A library for privacy-preserving machine learning. GitHub, 2021. `https://github.com/OpenMined/TenSEAL`.

[4] The Matplotlib Community. Matplotlib documentation, 2024. `https://matplotlib.org/stable/contents.html`.

[5] The NumPy Community. Numpy documentation, 2024. `https://numpy.org/doc`.

[6] Craig Gentry. Fully homomorphic encryption using ideal lattices. *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 169–178, 2009.

[7] Microsoft SEAL Development Team. Microsoft seal (release 4.0). GitHub, 2021. `https://github.com/microsoft/SEAL`.