Maturing Homomorphic Encryption (HE) to Enable Privacy Preserving Vector Search

Debabrata Auddya [1], Lander Besabe [2], Marina Chugunova [3], Sandra Moreno Cristobal [4], Xinyi Hu [5], Chiu-Yen Kao [6], Maria Camila Mejia [7], Richard Moore [8], Theeraphat Ton Pothisawang [12], Reza Rassool [9], Sulimon Sattari [10], Kriti Sehgal [11]

[1] *auddya@udel.edu, University of Delaware*
[2] *lybesabe@central.uh.edu, Department of Mathematics*
[3] *marina.chugunova@cgu.edu, Claremont Graduate University*
[4] *sandramoreno@cpp.edu, California State Polytechnic University, Pomona*
[5] *xhu3914@sdsu.edu, San Diego State University*
[6] *ckao@cmc.edu, Claremont McKenna College, Claremont*
[7] *mariacamila.mejiagarcia01@utrgv.edu, University of Texas Rio Grande Valley*
[8] *moore@siam.org, Society for Industrial and Applied Mathematics*
[12] *theeraphat.pothisawang@cgu.edu, Claremont Graduate University*
[9] *reza@kwaai.ai, Kwaai, nonprofit AI Lab*
[10] *sulimon@kwaai.ai, Kwaai, nonprofit AI Lab*
[11] *ksehgal@uchicago.edu, The University of Chicago*

(*Communicated to* MIIR *on 28 October 2025*)

Study Group: Mathematical Problems in Industry Workshop, Claremont Graduate University, June 9–13, 2025

Communicated by: Marina Chugunova, Claremont Graduate University

Industrial Partner: Kwaai, nonprofit AI Lab

Presenter: Reza Rassool

Industrial Sector: AI Research

Tools: Python

Key Words: Homomorphic Encryption;

Contents

1 Abstract

As language models increasingly rely on vector embeddings stored on remote servers, safeguarding sensitive query data has become critically important. Homomorphic encryption (HE) enables computations on encrypted data, offering a search solution that preserves privacy. This paper reviews established HE techniques—including dimensional scrambling, noise injection, ElGamal, exponential ElGamal, CKKS, and chaotic mapping—and introduces two novel algorithms: DIEHARD and ROME. These methods are evaluated for their ability to preserve inner product operations, encryption strength, and computational efficiency. Our findings highlight ROME as a promising approach that balances security and performance while maintaining search accuracy.

2 Introduction

As the use of embeddings for language models grows, vector databases stored and queried on remote servers introduce new cybersecurity vulnerabilities. For example, queries against the vector database may contain sensitive information (e.g. health records), which could be retrieved via cyber attack. Can embeddings containing sensitive information be encrypted before they are uploaded to an untrusted host, such that the data is still searchable in its scrambled form? The type of encryption that allows for calculations to be performed in the encrypted space is known as homomorphic encryption, which was first proposed as a tool by Rivest et. al [8]. The first fully homomorphic encryption technique, which could be used to perform any calculation that could be performed on logic gates, was proposed by Craig Gentry by using ideal lattices 30 years later [4].

Consider the ciphertexts c_i decrypt to plaintext m_i for $1 \leq i \leq n$, i.e., Decrypt(c_i)= m_i where m_i's and c_i's are elements of some ring (with two operations, addition and multiplication) [9]. We say that an encryption is a fully Homomorphic Encryption if

$$\text{Decrypt}(c_1 + c_2) = m_1 + m_2, \quad \text{Decrypt}(c_1 \cdot c_2) = m_1 \cdot m_2.$$

Whenever these two operators are preserved, if f is a function composed of finitely many additions and multiplication in the ring, e.g., inner product, then

$$\text{Decrypt}(f(c_1, \cdots, c_n)) = f(m_1, \cdots, m_n).$$

Assume that Bob can connect to a large set of sensitive documents $m_1, \cdots, m_n$ and encrypts these to obtain ciphertexts $c_1, \cdots, c_n$. Bob sends ciphertexts to Eve who offers computational power to perform the calculation $f(c_1, \cdots, c_n)$. After the calculation, Eve sends the results back to Bob so he decrypts them to get $f(m_1, \cdots, m_n)$. As hackers usually have knowledge about encryption algorithms and may obtain information about some queries and computational results, it is important to design encryption algorithms which are robust even when the aforementioned information are available.

In this report, we first review some encryption algorithms including dimensional scrambling, noise injection method, ElGamal cryptosystem [3], exponential ElGamal cryptosystem [11], and CKKS scheme [2] in Section 3. See [1] for a survey article which includes some other homomorphic encryption methods. We then present a known algorithm: Chaotic Mapping, and two new algorithms: DIEHARD and ROME in Section 4.

We implement these encryption methods and compare their efficiency. The paper ends with a brief conclusion on our findings and a discussion on potential future work in Section 7.

3 Encryption Algorithms

In this section, we briefly review several early algorithms including fully homomorphic encryption and partial homomorphic encryption which only preserves either addition or multiplication.

3.1 Dimensional Scrambling

Let $A = \{\mathbf{a}_1, \cdots, \mathbf{a}_q\}$ be the set of queries and $B = \{\mathbf{b}_1, \cdots, \mathbf{b}_d\}$ be the set documents.

Given a vector $\mathbf{v} \in \mathbb{R}^n$, the dimensional scrambling [12] refers to a permutation of elements in $\mathbf{v}$. Denote $\xi(\mathbf{v})$ as the permuted $\mathbf{v}$. We know that the inner product of two vectors $\mathbf{a}$ and $\mathbf{b}$ is preserved:

$$\xi(\mathbf{a}) \cdot \xi(\mathbf{b}) = \mathbf{a} \cdot \mathbf{b}.$$

This is equivalent to generating a random permutation matrix $\mathcal{Q} \in \mathbb{R}^{n \times n}$ and definition $\xi(\mathbf{a}) = \mathcal{Q}\mathbf{a}$. Note that this preserves the inner product since $\mathcal{Q}$ is orthogonal:

$$\xi(\mathbf{a}) \cdot \xi(\mathbf{b}) = (\mathcal{Q}\mathbf{a})^T(\mathcal{Q}\mathbf{b}) = \mathbf{a}^T\mathcal{Q}^T\mathcal{Q}\mathbf{b} = \mathbf{a}^T\mathbf{b} = \mathbf{a} \cdot \mathbf{b}.$$

3.2 Noise Injection

Consider two noise vectors whose element-wise product is unity:

$$\hat{\eta} = (\eta_1, \eta_2, \cdots, \eta_n) \quad \text{and} \quad \check{\eta} = \left(\frac{1}{\eta_1}, \frac{1}{\eta_2}, \cdots, \frac{1}{\eta_n}\right).$$

Let the vectors $\mathbf{a} = \{a_i\}_{1 \le i \le n}$ and $\mathbf{b} = \{b_i\}_{1 \le i \le n} \in \mathbb{R}^n$ correspond to the embedded queries and documents, respectively. Define

$$\xi(\mathbf{a}) = \mathbf{a} \circ \hat{\eta} = \{a_i\eta_i\}_{1 \le i \le n}.$$

We can verify that the inner product is preserved after noise injection since

$$\xi(\mathbf{a})\xi(\mathbf{b}) = (\mathbf{a} \circ \hat{\eta}) \cdot (\mathbf{b} \circ \check{\eta}) = \{a_i\eta_i\}_{1 \le i \le n} \cdot \left\{\frac{b_i}{\eta_i}\right\}_{1 \le i \le n} = \sum_{i=1}^{n} a_i\eta_i b_i \frac{1}{\eta_i} = \mathbf{a} \cdot \mathbf{b}.$$

To implement noise injection into the encryption algorithm, we generate the noise vector as a secret key and incorporate it into our encrypted queries and documents $\xi(\mathbf{a})$ and $\xi(\mathbf{b})$ by element-wise multiplying them with the noise. This approach secures the privacy of user data while maintaining the accuracy of the search results.

3.3 ElGamal Cryptosystem

The ElGamal encryption system, introduced by Taher El Gamal in 1985, is an asymmetric cryptographic algorithm based on the Diffie-Hellman key exchange between two parties to encrypt a message. This algorithm is based on the difficulty of computing discrete logarithms in a cyclic group.

Taher ElGamal introduced the ElGamal public-key cryptosystem in 1985 [3]. The key generation steps include

- Choose a large prime number p and a generator g of the multiplicative group $\mathbb{Z}_p^{\otimes}$.
- Select a generator g.

The ElGamal cryptosystem [3] preserves the multiplication while the exponential ElGamal cryptosystem preserves the addition.

3.4 Exponential ElGamal Cryptosystem

The exponential ElGamal encryption system is an enhanced variant of the original ElGamal scheme, as introduced by Zhou et al. [11]. While ElGamal is recognized as one of the earliest homomorphic encryption protocols and is known for supporting multiplicative homomorphism, it also has notable limitations. In particular, attackers can utilize the relationship between the quadratic residue properties of plaintexts and their ciphertexts, exposing the scheme to vulnerabilities such as chosen plaintext attacks. Therefore, the exponential ElGamal scheme, is proposed to resist the vulnerabilities of the existing encryption protocol. The following algorithm describes the additive homomorphism which is a characteristic of this scheme.

The input to this algorithm is similar to that of the El Gamal in which the input is given as a small integer m, and a public key and a randomness number. Contrary to the standard El Gamal, where the m is chosen as a group member, in the exponential one, m can be any small integer. The procedure is outlined as follows:

Similar to the standard El Gamal method, generate a large prime number p, a generator g and a random private key x such that $x \in \{1, 2, \ldots, p-2\}$. The public key is computed as $h = g^x \bmod p$. For the encryption part, a random number $r \in \{1, 2, \ldots, p-2\}$ is chosen, followed by computing the ciphertext tuple which is represented by (c_1, c_2). The components of this tuple are given by:

$$c_1 = g^r \bmod p \qquad c_2 = g^m h^r \bmod p$$

When encrypting the sender uses the receiver's public key ($h = g^x$) and a random number r to create part of the ciphertext. Given this tuple, the shared key is given as:

$$s = c_1^x \bmod p = (g^r)^x = g^{rx} \bmod p$$

Following this step, the masking component attached to c_2 is removed by

$$\frac{c_2}{s} \bmod p = \frac{g^m h^r}{g^{xr}} \bmod p = g_m$$

since $h = g^x$. Once g_m is obtained, m is recovered by computing the discrete logarithmic base g of g_m.

Once decryption is complete and the original message has been recovered, it is also possible to take advantage of exponential ElGamal's homomorphic property. To demonstrate it, a separate ciphertext of the form (d_1, d_2) is considered, so that

$$g^{r_1} g^{r_2} = g^{r_1+r_2} \quad \text{and} \quad (g^{m_1} h^{r_1})(g^{m_2} h^{r_2}) = g^{m_1+m_2} h^{r_1+r_2}$$

This generates a new ciphertext that encrypts $m_1 + m_2$ with randomness $r_1 + r_2$.

3.5 CKKS Homomorphic Encryption Scheme

The Cheon-Kim-Kim-Song (CKKS) encryption [2] is an approximate homomorphic encryption scheme, based on the learning with error problem [7], which allows for arithmetic in the complex space, and thus, the real space. This scheme exploits the structure of integer polynomial rings. The idea is to encode the message $\mathbf{a} \in \mathbb{C}^n$ into a polynomial $m(X) \in \mathbb{C}[X]/(X^n + 1)$ which we will use the canonical embedding $\sigma : \mathbb{C}[X]/(X^n + 1) \to \mathbb{C}^n$ to encode and decode the messages and polynomials, respectively.

Note that the embedding σ is an isomorphism, i.e., σ is a bijective homomorphism and thus, any vector $\mathbf{a}$ is encoded into a unique corresponding polynomial in $\mathbb{C}[X]/(X^n+1)$, and vice versa.

To decode a polynomial $m(X)$ into a vector $\mathbf{a} \in \mathbb{C}^n$, we evaluate the polynomial m on the n roots of the M-th cyclotomic polynomial $\Phi_M(X) = X^M + 1$, where $n = 2M$, which are $\varphi, \varphi^3, \ldots, \varphi^{2n-1}$:

$$z = \sigma(m) = \left(m(\varphi), m(\varphi^3), \ldots, m(\varphi^{2n-1})\right) \in \mathbb{C}^n.$$

Now, to encode $\mathbf{a}$, we need to compute the inverse σ^{-1}. The encoding problem reads: find a polynomial $m(X) = \sum_{i=0}^{n-1} \alpha_i X^i \in \mathbb{C}[X]/(X^n + 1)$, given $\mathbf{a} \in \mathbb{C}^n$ such that $\sigma(m) = (m(\varphi), m(\varphi^3), \ldots, m(\varphi^{2n-1})) = (a_1, a_2, \cdots, a_n)$. This is equivalent to solving the linear system $\mathcal{A}\alpha = \mathbf{a}$ where $\mathcal{A}_{i,j} = \left(\varphi^{2i-1}\right)^j$ is the Vandermonde matrix.

After encoding the message through the polynomial m, we choose a large scaling factor Δ and a large prime q to compute $\hat{\alpha}_i = \text{round}(\Delta\alpha_i) \in \mathbb{Z}$ and set $\hat{\alpha}_i \mapsto \hat{\alpha}_i \bmod q$.

To perform the encryption and decryption process, we randomly generate a secret key $\mathbf{s} \in \mathbb{Z}_q^n$, and uniformly sampled vectors $\mathbf{m}_i \in \mathbb{Z}_q^n$. We publish the public key $\mathbf{p}$ defined as

$$\mathbf{p} = (-\mathcal{A}\mathbf{s} + \mathbf{e}, \mathcal{A})$$

where $\mathbf{e} \in \mathbb{Z}_q^n$ is a small error vector (typically Gaussian). Due to the small error we introduced, the secret key $\mathbf{s}$ will be difficult to figure out. Else, this could be solved using Gaussian elimination.

Now, let $\mathbf{m} \in \mathbb{Z}_q^n$ to be the encoded message we want to encrypt and then decrypt. The encryption $\mathbf{c}$ of $\mathbf{m}$ is given by

$$\mathbf{c} = (\mathbf{m}, \mathbf{0}) + \mathbf{p} = (\mathbf{m} - \mathcal{A}\mathbf{s} + \mathbf{e}, \mathcal{A}) = (\mathbf{c}_0, \mathbf{c}_1)\,.$$

To decrypt $\mathbf{c}$ back into $\mathbf{m}$, we perform the following operations:

$$\mathbf{c}_0 + \mathbf{c}_1\mathbf{s} = \mathbf{m} - \mathcal{A}\mathbf{s} + \mathbf{e} + \mathcal{A}\mathbf{s} = \mathbf{m} + \mathbf{e} \approx \mathbf{m}$$

for sufficiently small $\mathbf{e}$.

3.6 Chaotic Mapping

Including a chaotic sequence into the encryption algorithm is another alternative approach. A chaotic sequence is typically generated by using a chaotic map. This type of map generates a pseudo-random sequence by starting with a set of initial conditions. Many chaotic sequences are generated from nonlinear maps, so their mapping relations are irreversible.

In this project, a chaotic sequence $\mathbf{y} = [y_1, y_2, \ldots, y_n]$ is generated using an improved logistic map defined by $y_n = 1 - 2y_{n-1}^2$ given y_0. This improved map is first proposed in [5] ensures that all the values of the sequence are between 1 and -1, i.e., $(-1 \leq y_n \geq 1)$. By using any initial condition that is in the basin of chaotic attractor of this map, we generate a chaotic sequence and combine it with a random number β so that the encryption becomes $\xi(\mathbf{a}) = \mathbf{a} + \beta\mathbf{y}$ and $\xi(\mathbf{b}) = \mathbf{b} + \beta\mathbf{y}$.

The simplest implementation of this encryption scheme does not preserve neither additive nor multiplicative homomorphism. An improvement the team has developed to approximately preserve the scalar product is to include asymmetric chaotic encoding such that the query is encrypted as $\xi(\mathbf{a}) = \mathbf{a} + \beta\mathbf{y}$ and the document is encrypted as $\xi(\mathbf{b}) = \mathbf{b} - \beta\mathbf{y}$.

It is straightforward to verify that **asymmetric chaotic encoding preserves additive homomorphism**, as the transformation satisfies $\xi(\mathbf{a}) + \xi(\mathbf{b}) = \mathbf{a} + \mathbf{b}$. However, preserving the dot product under chaotic transformations is more challenging. Note that $\xi(\mathbf{a}) \cdot \xi(\mathbf{b}) = AB + \beta\mathbf{y}(B - A) - (\beta\mathbf{y})^2)$. To preserve the dot product, we want to restrict the correction term to be approximately zero.

This requires either imposing specific constraints on the chaotic mapping or integrating modular arithmetic into the encryption scheme to control cross-terms and maintain inner product structure.

Although chaotic mapping encryption schemes offer strong security and low computational overhead, they do not inherently preserve the dot product, which is critical for tasks like similarity search or secure retrieval. Future research should aim to develop asymmetric or structure-preserving transformations that maintain inner product relationships while retaining the benefits of chaos-based encryption.

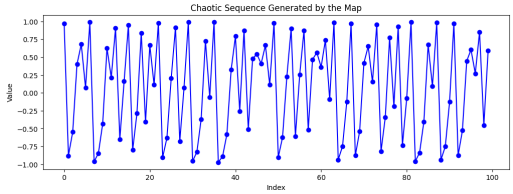

Figure 1. The first 100th iteration of the chaotic sequence is generated by the improved chaotic map

4 New Algorithms

4.1 Dimension-Increasing Encrypting Homomorphism Algorithm for Randomized Discovery (DIEHARD)

The three methods above, dimensional scrambling, noise injection, and random extended orthogonal encryption, are all limiting cases of general linear encryption methods that preserve the inner product of a query vector with a document vector. To see this, consider query encryption of the form

$$\tilde{\mathbf{a}} = \mathcal{A}\mathbf{a}, \quad \tilde{\mathbf{b}} = \mathcal{B}\mathbf{b},$$

where $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{m \times n}$, so that encryption maps n-vectors to m-vectors. Since

$$\tilde{\mathbf{a}} \cdot \tilde{\mathbf{b}} = (\mathcal{A}\mathbf{a})^T \mathcal{B}\mathbf{b} = \mathbf{a}^T \mathcal{A}^T \mathcal{B}\mathbf{b},$$

reservation of the inner product requires

$$\mathcal{A}^T \mathcal{B} = I_n,$$

where I_n is the $n \times n$ identity matrix. Note that this assumes $m \geq n$.

Forming the query-document encryption pair is straightforward. Create a random $m \times n$ matrix $\mathcal{A}$ (where the distribution is unimportant but Gaussian and uniform provide convenient choices). Verify that $\mathcal{A}$ is full rank, e.g., by verifying that

$$\det \mathcal{A}^T \mathcal{A} > 0.$$

If not, redraw. If so, set

$$\mathcal{B} = \mathcal{A}(\mathcal{A}^T \mathcal{A})^{-1}.$$

Learning the query encryption matrix $\mathcal{A}$ from a set of q queries requires solving for mn entries from mq equations. Clearly, $q > n$ is required for this to be well-posed. Thus, choosing the size of encrypted query vectors to be larger than the size of the plain-text query vectors offers no significant advantage in terms of encryption strength.

A more serious drawback to this approach is the fact that neither $\mathcal{A}$ nor $\mathcal{B}$ preserves the query and document vector norms under encryption. Adding those conditions requires

$$\mathcal{A}^T \mathcal{A} = \mathcal{B}^T \mathcal{B} = I_n$$

in addition to the requirement that $\mathcal{A}^T B = I_n$. This is only possible if $\mathcal{B} = \mathcal{A}$.

To see why this is true, note that, without loss of generality, $\mathcal{B}$ can be written in the form

$$\mathcal{B} = \mathcal{A} + \mathcal{A}_1,$$

where $\mathcal{A}_1$ is in the nullspace of $\mathcal{A}^T$. But then

$$\mathcal{B}^T \mathcal{B} = \mathcal{A}^T \mathcal{A} + \mathcal{A}_1^T \mathcal{A}_1 = I_n + \mathcal{A}_1^T \mathcal{A}_1.$$

Thus $\mathcal{A}_1^T \mathcal{A}_1 = 0$, which implies that all columns of $\mathcal{A}_1$ are vectors of norm zero, i.e., $\mathcal{A}_1 = 0$.

The queries and documents are therefore encrypted using the same norm-preserving matrix, whose action is restricted to rotations and zero-padding. This approach is discussed in the next section.

4.2 Random Orthogonal Method of Encryption (ROME)

Consider a message that is embedded as a vector $\mathbf{q} \in \mathbb{R}^n$ with $||\mathbf{q}|| = 1$. For example, n can be 1536 which is the length of the embedding vector for *text-embedding-ada-002* (most commonly used embedding model by OpenAI). In the ROME method, we perform the encryption in two steps. As a first step, we insert a fixed number of zeros (let us say Z) at random positions within the elements of $\mathbf{q}$ which generates a vector $E(\mathbf{q}) \in \mathbb{R}^m$ (where $m = n + Z$), e.g.

$$E(\mathbf{q}) = [q_1, 0, 0, q_2, q_3, \cdots, 0, 0, 0, q_4, \cdots, q_N, 0]^T$$

This step of adding zeros to extend the vector, we call, the zero-padding. We can think of doing this step by partitioning the number Z into $n + 1$ non-negative numbers such that the sum of those numbers is Z. The numbers in the partition will be the number of zeros that we insert between two elements of $\mathbf{q}$ starting from the left of the first element and ending at the right of the last element ($n + 1$ spaces). It is equivalent to multiplying $\mathbf{q}$ by a $m \times n$ matrix $\mathcal{E}$ of zeros and ones, where ones are placed such that the elements of $\mathbf{q}$ appears in $E(\mathbf{q})$ at the appropriate places.

Secondly, we generate a random orthonormal matrix $\mathcal{Q}$ (i.e., $\mathcal{Q}^{-1} = \mathcal{Q}^T$) of size $m \times m$. This can be implemented by generating m random independent vectors and using the Gram-Schmidt algorithm to orthonormalize (QR decomposition of a random full rank matrix). Computational complexity of QR decomposition is $O(m^3)$: this signifies that as the size of the matrix (m) doubles, the number of operations required for QR decomposition increases by a factor of (2^3). Householder reflections method is a common algorithm for calculating QR decomposition, and its complexity is typically $O(m^3)$. It was found that using random orthogonal matrix is statistically defensible as it allows users to make inferences about parameters in a model similar to raw dataset [10].

Assume $\mathbf{q}_1$ and $\mathbf{q}_2$ are two messages, one can easily verify that this random extended (zero-padding) orthogonal encryption preserves the inner product.

$$(\mathcal{Q}\mathcal{E}\mathbf{q_1}) \cdot (\mathcal{Q}\mathcal{E}\mathbf{q_2}) = \mathbf{q_1} \cdot \mathbf{q_2}.$$

Therefore, ROME is a homomorphic encryption method.

4.2.1 *Minimum number of queries to know to hack ROME*

To decrypt the ROME method, we need to obtain the matrix $\mathcal{Q}\mathcal{E}$ i.e. we have mn unknowns.

4.2.2 *Hacking ROME- 2×2 case*

In this section, we show our attempt to hacking into the ROME algorithm for a simple case. We start with a query of size $\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$ To encrypt, we do not perform zero padding and simply multiply $\mathbf{q}$ with the orthonormal matrix

$$\mathcal{Q} = \begin{bmatrix} \eta_{11} & \eta_{12} \\ \eta_{21} & \eta_{22} \end{bmatrix}$$

to get the encrypted query vector $E(\mathbf{q}) = \mathcal{Q}\mathcal{E}\mathbf{q} = \begin{bmatrix}\tilde{q_1}\\ \tilde{q_2}\end{bmatrix}$.

We assume that the hacker has access to $\mathbf{q}$ and $E(\mathbf{q})$ and our goal is to find the matrix $\mathcal{Q}$. We assume that the hacker knows that the encryption is homomorphic and therefore $\mathcal{Q}^T\mathcal{Q} = \mathcal{I}$. Therefore, the hacker has the following equations:

$$\eta_{11}^2 + \eta_{21}^2 = 1,\ \eta_{12}^2 + \eta_{22}^2 = 1,\ \eta_{11}\eta_{12} + \eta_{21}\eta_{22} = 0,$$
$$\eta_{11}q_1 + \eta_{12}q_2 = \tilde{q_1},\ \eta_{21}q_1 + \eta_{22}q_2 = \tilde{q_2}$$

The solution for the above problem is obtained by solving the following quadratic equations for η_{11} and η_{22} (respectively):

$$(\tilde{q_1}^2 + \tilde{q_2}^2)\eta_{11}^2 - 2q_1\tilde{q_1}\eta_{11} + (q_1^2 - \tilde{q_2}^2) = 0$$
$$(\tilde{q_1}^2 + \tilde{q_2}^2)\eta_{22}^2 - 2q_2\tilde{q_2}\eta_{22} + (q_2^2 - \tilde{q_1}^2) = 0$$

which can further be simplified by thinking of $\tilde{\mathbf{q}}$ as a unitary vector. Once we have the values for η_{11} and η_{22}, we find η_{21} and η_{12} by:

$$\eta_{21} = \frac{q_1 - \eta_{11}\tilde{q_1}}{\tilde{q_2}},\ \eta_{12} = \frac{q_2 - \eta_{22}\tilde{q_2}}{\tilde{q_2}}.$$

The above shows that the solutions tend to be non-unique even in the simplest case. The above also shows that only one query is not enough to know the exact matrix $\mathcal{Q}$ due to non-uniqueness of solutions.

Encryption Methods	Homomorphism	Strength	Computational Expense
Dimensional+Noise	Yes	Weak	26.24 μs $\pm$ 7.98 μs
El Gamal	Multiplication only	Strong	3.39 s $\pm$ 146 m
Exponential El Gamal	Addition only	Strong	4.66 s $\pm$ 218 ms
CKKS	Approximately	Strong	32.3 ms $\pm$ 640 ms
Chaos mapping	Approximately	Strong	17 μs $\pm$ 20 μs
ROME	Yes	Strong	8.51 ms $\pm$ 3.43 ms

Table 1. Summary of Encryption Techniques: Trade-offs Between Efficiency and Security

5 Implementation

In this section, we implemented all the algorithms described above in the previous section s. To begin, we created a small sample of textual documents, each comprising a simple sentence related to academia, programming, or artificial intelligence. These documents were then converted into numerical representations using the pre-trained SentenceTransformer model 'all-MiniLM-L6-v2'. This model produces dense vector embeddings for each sentence.

To simulate querying within an encrypted document retrieval system, we defined a set of user queries and saved them to a text file to ensure reproducibility. These queries were later loaded and embedded into dense vectors using the same pre-trained SentenceTransformer model. Both the unnormalized and normalized forms of the resulting query embeddings were saved for further analysis.

Finally, we applied each of the previously described methods to the document embeddings and the query embeddings to perform encryption. Table 1 shows a comparative analysis of all the encryption methods proposed based on the homomorphism properties, encryption strength and computation expense.

From the implemented methods we can notice that CKKS operates on data in a row-wise manner, which generally results in lower running time. ElGamal and Exponential ElGamal are less efficient in our implementation, as the available packages only support encrypting one number at a time. We did not attempt to optimize these implementations; instead, we followed the standard procedures described in the documentation. In addition, neither ElGamal nor Exponential ElGamal preserves the dot product property. Given that ElGamal and Exponential ElGamal were implemented in the simplest possible way and do not preserve the dot product, they are not directly comparable to the other methods in terms of performance or functionality.

Chaos mapping demonstrates one of the lowest computational times among the methods evaluated. However, it only preserves addition and not the dot product. Dimension plus noise injection and CKKS are among the methods with lower computational costs. However, while dimensional scrambling combined with noise injection does preserve the dot product, it is relatively weak in terms of security—meaning it is easier to decrypt. Although CKKS is considered a strong encryption method, it does not preserve the dot product property. In contrast, our newly proposed method, ROME, both preserves the dot product and offers strong security, with computational expense comparable to some of the more efficient methods.

6 Sacking ROME

ROME is vulnerable to an actor intercepting the unencrypted and encrypted queries and using them to infer $\mathcal{QE}$. Two intuitive methods of solution are to (1) account for the fact that $\mathcal{QE}$ is formed by zero-padding followed by the action of an orthogonal matrix and (2) simply solve for the entire $m \times n$ matrix. We assume for now that there are sufficiently many encrypted-unencrypted query pairs to implicitly enforce the orthogonality constraint on Q, so that each case can be solved using linear algebra.

The $m \times n$ entries of $\mathcal{QE}$ are determined by the $q \times m$ equations relating the matrix of q known encrypted query vectors, A, to the matrix of q known unencrypted queries, i.e.,

$$\tilde{A} = \mathcal{QE}A.$$

Assuming $q \geq n$, this is solved to give

$$\mathcal{QE}AA^T = \tilde{A}A^T.$$

The computational complexity of solving for $\mathcal{QE}$ includes:

- formation of the right-hand side: qmn
- formation of the matrix AA^T: qn^2
- solution by Gaussian elimination: $\frac{2}{3}n^3 + mn^2$

In total, this gives

$$C^q_{mn} = qmn + qn^2 + \frac{2}{3}n^3 + mn^2.$$

Alternatively, one could try solving for the $(n \times n)$-dimensional permutation matrix remaining after excluding $m - n$ entries of the query vector, repeating that process for each of the $\binom{m}{m-n}$ combinations of zero-padded vector entries. This gives

$$\hat{C}^q_{mn} = \frac{m!}{(m-n)!n!}C^q_{nn} = \frac{m!}{(m-n)!n!}(2qn^2 + \frac{5}{3}n^3).$$

As demonstrated in Fig. 2, the ratio of the computational cost for solving for $n \times n$ matrices over all zero-padding combinations to that for simply solving for the $m \times n$ matrix $\mathcal{QE}$ all at once is always greater than one, often considerably so. Simply solving for the full matrix $\mathcal{QE}$ is therefore more efficient.

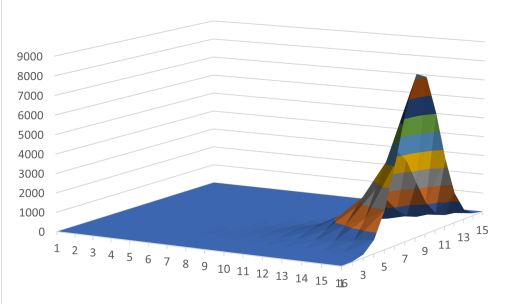

Figure 2. Ratio of computational cost, $m\mathrm{C}(m-n)$ solves of n^2 vs single solve of $m \times n$ (for n queries)

7 Conclusion and Future Work

Homomorphic encryption presents a promising direction to securely outsourcing computations on encrypted data, ensuring privacy even in the presence of third-party computation, like the distributed computing system that Kwaai offers. In this work, we review various existing encryption algorithms: dimensional scrambling, noise injection, chaos mapping, ElGamal cryptosystem, and CKKS on their capability to preserve the scalar product, strength, and computational efficiency. In addition, we introduced two homomorphic encryption algorithms, dimension-increasing encrypting algorithm for randomized discovery (DIEHARD) and randomized orthogonal method of encryption (ROME).

In addition to the results achieved this week, the team would like to continue investigating how to improve the VPR used to homomorphically encrypt Personal AI [6]. Some of the encryptions that should be explored in greater depth are CKKS (to improve its accuracy) and lattice encryption (as an alternative homomorphic encryption algorithm).

Another improvement that should be made lies in ROME. It would be ideal to create an algorithm as strong as ROME that does not expand the data when implemented.

Finally, we recommend exploring the use of multiple encryption devices, each requiring a different key to be accessed, thus complicating the efforts of a malicious actor to obtain sensitive information.

References

[1]Abbas Acar et al. "A survey on homomorphic encryption schemes: Theory and implementation". In: *ACM Computing Surveys (Csur)* 51.4 (2018), pp. 1–35.

[2]Jung Hee Cheon et al. "Homomorphic encryption for arithmetic of approximate numbers". In: *Advances in Cryptology – ASIACRYPT 2017*. Lecture notes in computer science. Cham: Springer International Publishing, 2017, pp. 409–437.

[3]Taher ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE transactions on information theory* 31.4 (1985), pp. 469–472.

[4]Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 169–178.

[5]Zhenghan He and Weibin Zhang. "Research on real time network homomorphic encryption algorithm based on chaotic sequence". In: (Nov. 2020), pp. 463–466. DOI: 10.1109/ICRIS52159.2020.00119.

[6]Kristin Lauter. "Private Artificial Intelligence: Machine Learning on Encrypted Data". In: *Collections* 55.03 (2022).

[7]Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". en. In: *J. ACM* 56.6 (Sept. 2009), pp. 1–40.

[8]Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. "On data banks and privacy homomorphisms". In: *Foundations of secure computation* 4.11 (1978), pp. 169–180.

[9]Alice Silverberg. "Fully homomorphic encryption for mathematicians". In: *Cryptology ePrint Archive* (2013).

[10]Daniel Ting, Stephen E Fienberg, and Mario Trottini. "Random orthogonal matrix masking methodology for microdata release". In: *International Journal of Information and Computer Security* 2.1 (2008), pp. 86–105.

[11]Ronghao Zhou and Zijing Lin. "An improved exponential elgamal encryption scheme with additive homomorphism". In: *2022 International Conference on Blockchain Technology and Information Security (ICBCTIS)*. IEEE. 2022, pp. 25–27.

[12]Hai-Hua Zhu, Zi-Gang Chen, and Tao Leng. "Random permutation-based mixed-double scrambling technique for encrypting MQIR image". en. In: *J. Appl. Phys.* 135.1 (Jan. 2024).