HyDia: FHE-based Facial Matching with Hybrid Approximations and Diagonalization

Samuel Martin, Nirajan Koirala, Helena Berens, Thomas Rozgonyi, Micah Brody, Taeho Jung
University of Notre Dame
{smarti39,nkoirala,hberens,trozgony,mbrody,tjung}@nd.edu

Abstract

Secure facial matching systems play a crucial role in privacy preserving biometric authentication, particularly in domains such as law enforcement, border control, and healthcare. Traditional facial matching systems require direct access to biometric data, raising significant privacy concerns. This paper presents HyDia, a novel protocol for scalable FHE-based facial matching with high computation and communication efficiencies, enabling secure one-to-many facial matching without exposing biometric data in plaintext. Our protocol adapts diagonalized matrix multiplication techniques to accommodate highly imbalanced matrix computations, enabling our novel non-rotational inner product algorithm that substantially reduces the homomorphic computation overhead compared to prior works. We further propose a hybrid approximation method for homomorphic thresholding, which achieves better approximation than the state-of-the-art approach (Chebyshev approximation) at the same multiplicative depths. More importantly, our design does not reveal exact similarity scores to the querier; instead, it provides only a threshold-based match decision or matching sources, strengthening privacy by withholding granular database information. We implement HyDia and competing approaches and provide both formal security proof and extensive experimental validation. Our results show that HyDia achieves practical query times at scale, significantly outperforming existing HE-based solutions in both computation and communication overhead. Notably, HyDia is the only viable FHE-based approach in common bandwidth settings (2Mbps & 1Gbps), outperforming the state-of-the-art approaches by 5.2×-227.4× in end-to-end latency under different settings. Finally, our experiments on real-face datasets show that HyDia incurs negligible accuracy loss, by achieving the same F_1 score of 0.9968 as the corresponding plaintext facial matching baselines. This work advances the feasibility of privacy-preserving biometric identification, offering a scalable, bandwidth-efficient, and accurate solution for real-world deployments.

Keywords

Facial Matching, Biometric Security, FHE, CKKS

1 Introduction

Facial matching technologies are becoming increasingly prevalent across fields such as law enforcement, healthcare, and banking. However, biometric identifiers like facial images used in these technologies are classified as personally identifiable information (PII) and subject to strict regulations on their storage and dissemination [49]. Moreover, facial matching systems that perform biometric matching using unencrypted data render that data vulnerable to attacks [51], highlighting the need for a facial matching system that operates entirely on encrypted data [5, 16, 19, 29].

In one case, a party would use facial matching to learn whether a subject is a member of, e.g., a watch list of known or suspected individuals [67], which we call the *Membership scenario*. For example, the authorities at the border or airport customs checkpoint regularly employ facial matching technologies to determine whether passengers appear in any watchlist datasets [50]. In another case, a party would use facial matching to determine the specific identity of a subject for the purposes of verification or authentication, which we call *Identification scenario*. For example, healthcare facilities often employ facial matching technologies to identify patients when they leave the facility and to track whether specific patients have taken their required medications in time [47]. In both scenarios, due to privacy concerns and regulations [7, 22, 47, 68], facial data must be protected so that they do not transmit the PII and/or violate individuals' privacy. Thus, achieving such functionality of facial matching technologies while abiding by regulations requires a facial matching system that can compare a query containing encrypted facial data to a database of facial datasets without ever revealing the plain facial information or identifiers. These examples highlight the emerging need for a facial matching system in which both subject and database biometrics remain encrypted throughout the computation. In such cases, large databases with hundreds of thousands of entries require one to efficiently verify a subject's enrollment status or confirm their identity within the database. Hence, in time-sensitive environments (e.g., borders, customs, healthcare), fast facial matching is crucial for maintaining safety, making low-latency query processing critical.

Fully Homomorphic Encryption (FHE) is widely used in privacy-preserving facial recognition due to its low communication overhead (single-round communication and lower communication sizes compared to other interactive protocols) and strong security. This work focuses on FHE-based secure facial matching for its non-interactivity and robustness. Unlike multi-round protocols (e.g., secure multiparty computation or oblivious transfer) that suffer from high latency, recent FHE advancements enable efficient computations, making FHE-based facial matching practical while providing provable security guarantees against quantum-capable attackers.

Our approach follows the standard FHE-based framework as outlined in Figure 1 (the entities Client $\mathcal{C}$, Enroller $\mathcal{E}$, and Server $\mathcal{S}$ are defined in Section 4.1). More formally, we define two different scenarios of the one-to-many facial matching problem, *Membership*

Proceedings on Privacy Enhancing Technologies 2025(4), 585–604

https://doi.org/10.56553/popets-2025-0146

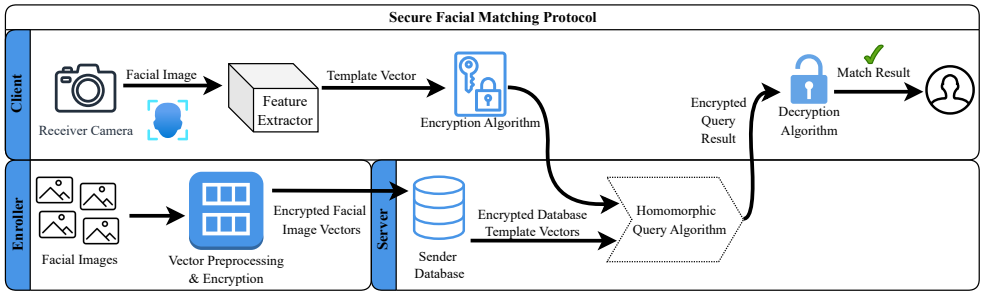

Figure 1: A high-level FHE-based facial matching protocol

scenario and *Identification scenario*. In the membership scenario, a client possesses an image of a subject's face and wants to learn whether or not their subject matches any images held by a server. The result of the membership scenario is a simple true or false statement given to the client by the server. In the identification scenario, a client still possesses an image of a subject's face, but now the client wishes to learn the exact indices of any matching images in the server's database. The result of the identification scenario is a list of indices of matching sources given to the client by the server, which the client can use to learn the identity of their subject, e.g., with an independent mapping reference.

Existing biometric recognition systems follow a common framework for one-to-many facial matching [5, 16, 19]. The client first extracts a real-valued facial feature vector (embedding) from an image, which the server then compares against all stored embeddings using a similarity metric, typically cosine similarity [5, 16, 17, 19, 29]. Existing FHE-based systems often perform a vector-preprocessing step before computing this similarity metric, to make computation more efficient [16, 19, 29]. The server then determines a match by thresholding similarity scores and returns either a binary result or matching indices. However, existing approaches [5, 16, 29] suffer from high computation overhead, making large-scale deployments where a single template must be matched against hundreds or thousands of templates impractical. Many methods compute homomorphic cosine similarity using rotation-based summation after ciphertext multiplication [5]. Homomorphic multiplication produces non-linear ciphertexts, preventing direct rotation, a crucial step in existing protocols. To compensate, relinearization is required, but its high computational cost severely impacts performance. As database size increases, this overhead becomes prohibitive; our experiments showed that querying 32,768 subjects resulted in an impractical four-minute delay, highlighting the inefficiency of these approaches in reducing latency.

Existing works, such as GROTE [29] and Blind-Match [16], reduce server overhead through various optimizations and reduce the number of ciphertext rotations (detailed below and in Section 3). However, these methods do not eliminate the primary bottleneck, i.e., the costly relinearization operation, which remains a major source of overhead. Recently, HERS [19] introduced a stacked encoding scheme, fully eliminating rotations in similarity computation and achieving state-of-the-art per-query server efficiency. However, its design still incurs unnecessary relinearization operations and requires the client to transmit large amounts of encrypted data for each query, making it impractical with network bandwidths $\leq$1Gbps. In conclusion, prior approaches exhibit efficiency in either computation or communication, but not both. Both are important for achieving low end-to-end overhead.

Moreover, in HERS and all other approaches except GROTE, the server reveals the exact cosine similarity scores for each database vector, allowing the client to gauge precisely how close non-matching entries are to the threshold. Similarity scores of feature descriptors or facial templates can be used to reconstruct images [39, 70], and recent deep generative models can successfully infer faces from face recognition systems [36]. Therefore, individual privacy will be at risk if similarity scores are disclosed. GROTE obscures similarity scores but assumes the server's dataset contains at most one match per query. By withholding exact cosine similarity scores, our work prevents adversarial inferences about the server's database distribution and provides better privacy without compromising accuracy or efficiency.

We propose a novel FHE-based facial matching protocol, HyDia, for one-to-many face matching. All parties are assumed to be semi-honest, and they operate on homomorphically encrypted data, thus eliminating the need to store unencrypted biometric data on the server. HyDia significantly improves both computation and communication efficiencies of state-of-the-art FHE-based approaches while improving their privacy guarantees. We adapt the diagonal matrix encoding for square matrices to our scenario, requiring highly imbalanced matrix multiplications. This encoding eliminates unnecessary rotations and relinearizations during cosine similarity computation, removes computation dependencies enabling parallel computing, and minimizes per-query communication overhead. Furthermore, we use homomorphic thresholding to hide similarity scores. This prevents the client from gauging a non-matching template's proximity to the threshold, ensuring stronger privacy than methods that expose full or partial similarity scores [16, 19] and thus risk the score-based face reconstruction attacks [39]. To ensure the homomorphic thresholding incurs negligible accuracy loss and additional overhead, we combine two distinct approximation methods to reduce approximation errors while maintaining low multiplicative depths of FHE. As a result, our protocol scales to databases with up to one million facial templates with the best computation and communication overhead with negligible accuracy loss. Notably, ours is the first to achieve FHE-based one-to-many face matching with low end-to-end latencies in practical network settings (<10s with 1Gbps and <100s with 2Mbps), whereas others are 5.2×-227.4× slower.

The contributions of this work are summarized as follows:

Table 1: List of Notations

Notation	Description
N	Scheme ring dimension, must be power of 2
$\Phi_M(X) = x^N + 1$	Cyclotomic polynomial of order $M = 2N$
$\mathcal{R} = \mathbb{Z}[X]/\langle\Phi_M(X)\rangle$	Polynomial ring
$\mathcal{E}$	The Enroller party
C	The Client party
$\mathcal{S}$	The Server party
ℓ	Dimension of template vectors, must be power of 2
K	Number of database template vectors
$D = \{\vec{d}_0, ..., \vec{d}_{K-1}\}$	Set of database template vectors
$\vec{q}$	Query template vector
$\theta \in [-1, 1]$	Cosine similarity match threshold
κ	Multiplicative depth allotted to threshold function
λ	Computational security parameter
$\mathbf{M}$	Set of ciphertexts storing encrypted database vectors
$\mathbf{q}$	Ciphertext storing encrypted query vector

- We adapt the commonly-used diagonal square matrix encoding [26] for the facial matching scenario requiring highly imbalanced matrix multiplications. HyDia supports efficient parallel-friendly protocols with single-ciphertext queries, achieving efficiency in both computation and communication.
- We combine different approximation methods to achieve a hybrid approximation with up to 3.17× speedup and better approximation fidelity compared to state-of-the-art approximation methods. Consequently, HyDia introduces negligible accuracy loss even with homomorphic thresholding that conceals similarity scores.
- We implement HyDia and other state-of-the-art approaches for rigorous evaluation. The source codes are released for reproducibility at https://github.com/n7koirala/image_matching/. Extensive experiments using real-world and synthetic datasets confirm HyDia outperforms state-of-the-art FHE-based approaches with minimum overhead in both computation and communication with negligible accuracy loss.

2 Preliminaries

2.1 Notation

We use the notations specified in Table 1 throughout this paper. In all algorithmic and protocol descriptions, an overhead arrow (e.g., $\vec{a}$) denotes an unencrypted real-valued vector, an overline (e.g., $\overline{a}$) represents a unit-length normalized variable, and boldface (e.g., $\mathbf{a}$) denotes a homomorphically encrypted ciphertext.

2.2 Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) is an encryption scheme that allows computations to be performed directly on encrypted data without access to the secret key. Since Gentry's introduction of the first practical FHE construction in 2009 [23], several FHE schemes have emerged, including BGV [10], BFV [9, 21], CKKS [12], and TFHE [15]. For improved performance, encryption parameters are typically set to support only circuits of a certain bounded multiplicative depth (leveled FHE), which we use in our protocol. The Cheon-Kim-Kim-Song (CKKS) scheme excels in real-number arithmetic applications like privacy-preserving facial matching, as it supports approximate arithmetic on encrypted floating-point values rather than only on finite-field operations. CKKS operates on elements of polynomial ring $\mathcal{R} = \mathbb{Z}[X]/\langle\Phi_M(X)\rangle$, where $\Phi_M(X)$ is the cyclotomic polynomial $(x^N + 1)$ of order $M = 2N$ (cyclotomic index) and degree $N \in \mathbb{Z}$ is the ring dimension.

A CKKS scheme supporting public-key encryption consists of a tuple of the following probabilistic polynomial-time algorithms and operations on ciphertexts:

- $\textsc{KeyGen}(\lambda, d, params) \rightarrow (pk, evk, sk)$: Given the parameters $params$, KeyGen produces a public key pk, a private key sk and a public evaluation key evk for homomorphic evaluations.
- $\textsc{Encrypt}(pt, pk) \rightarrow \mathbf{ct}$: Given a public key pk and plaintext $pt \in \mathcal{R}$, encryption algorithm sample $u \leftarrow \chi_{\text{enc}}$ and $e_0, e_1 \leftarrow \chi_{\text{err}}$ and outputs $\mathbf{ct} = (c_0, c_1) \leftarrow u \cdot pk + (pt + e_0, e_1) \bmod q$.
- $\textsc{Decrypt}(\mathbf{ct}, sk) \rightarrow pt$: Given ciphertext $\mathbf{ct} = (c_0, c_1)$ in R_q^2 and secret key $sk \in R_q$, the decryption algorithm computes, $pt^* = \left[c_0 + c_1 \cdot sk\right]_q$.
- $\textsc{Add}(\mathbf{ct}^{(1)}, \mathbf{ct}^{(2)}) \rightarrow \mathbf{ct}^{(3)}$: The addition operation returns a new ciphertext $\mathbf{ct}^{(3)}$ representing the element-wise sum of all slots in $\mathbf{ct}^{(1)}$ and $\mathbf{ct}^{(1)}$. This operation is computed as, $[c_0, c_1]^{(1)} + [c_0, c_1]^{(2)} = [(c_0^{(1)} + c_0^{(2)}), (c_1^{(1)} + c_1^{(2)})]$
- $\textsc{Multiply}(\mathbf{ct}^{(1)}, \mathbf{ct}^{(2)}) \rightarrow \mathbf{ct}^{(3)}$: The multiplication operation returns a ciphertext $\mathbf{ct}^{(3)}$ representing the element-wise product of all slots of the ciphertexts $\mathbf{ct}^{(1)}$ and $\mathbf{ct}^{(1)}$. It is computed as, $[c_0, c_1]^{(1)} \cdot [c_0, c_1]^{(2)} = [c_0^{(1)} c_0^{(2)}, c_0^{(1)} c_1^{(2)} + c_1^{(1)} c_0^{(2)}, c_1^{(1)} c_1^{(2)}]$. However, this results in a ciphertext comprised of three polynomials, in $\mathcal{R}^3$ instead of $\mathcal{R}^2$, which can be decrypted using the key sk^2. In order to prevent the number of polynomials from growing during successive multiplications, and to allow for decryption with sk, Relinearize operation is used.
- $\textsc{Relinearize}(\mathbf{ct}') \rightarrow \mathbf{ct}$: The relinearization operation, also known as key switching, converts a ciphertext in $\mathcal{R}^3$ to an equivalent ciphertext in $\mathcal{R}^2$, allowing decryption using the original private key sk [1].
- $\textsc{Rescale}(\mathbf{ct}') \rightarrow \mathbf{ct}$: This operation shortens the lower bits in the plaintext after multiplication.
- $\textsc{Rotate}(\mathbf{ct}', i) \rightarrow \mathbf{ct}$: For an integer i, the rotation operation returns a new ciphertext $\mathbf{ct}$ representing the message encrypted by $\mathbf{ct}'$, shifted by i amount of slots [1]. For example, if $i = 1$ and $\mathbf{ct}'$ is an encryption of the message $m = [x_0, x_1, ...x_{N/2-1}]$, then the resulting ciphertext $\mathbf{ct}$ will be an encryption of the rotated message: $m \gg 1 = [x_{N/2-1}, x_0, x_1, ...x_{N/2-2}]$

SIMD. A key feature of lattice-based FHE, including CKKS, is its *Single Instruction Multiple Data* (SIMD) capability [61]. A plaintext polynomial in the ring $\mathcal{R} = \mathbb{Z}[X]/\langle X^N + 1\rangle$ with N-degree cyclotomic polynomial encodes $N/2$ complex values via the canonical embedding. Thus, a single homomorphic operation acts element-wise on all $N/2$ values in parallel, efficiently batching multiple entries into one ciphertext with $N/2$ slots.

2.3 Facial Feature Extractors

Facial feature extractors are models based on Deep Convolutional Neural Networks that map facial images to template vectors [17]. These feature extractors have been engineered to map facial images to template vectors that possess high similarity for images of the same subject, and low similarity for images of distinct subjects [63]. Therefore, facial feature extractors are crucially employed in the facial matching task to convert facial images into vectors

for comparison [5, 16, 29]. Among the state-of-the-art examples of facial feature extractors are CosFace [69], SphereFace [45], and ArcFace [17], all of which employ cosine similarity as the similarity metric to be used in identity classification [16]. Cosine similarity is often used in tasks involving vector matching, like facial recognition and text embedding similarity, because it compares only the direction of vectors, ignoring their magnitude. Note that, if $\vec{x}$ and $\vec{y}$ are vectors of dimension ℓ, their cosine similarity score is $\cos(\vec{x}, \vec{y}) = \sum_{i=0}^{\ell}(\vec{x}[i]\vec{y}[i])/(||\vec{x}||||\vec{y}||) = \sum_{i=0}^{\ell}(\overline{x}[i]\overline{y}[i])$ where $\overline{x} = \vec{x}/||\vec{x}||$ is the unit-length normalization of $\vec{x}$. Therefore, when both vectors are unit-length normalized, computing their cosine similarity is equivalent to their inner product. To determine whether two facial images match or not, we compare their template vectors' cosine similarity score and compare it against a match threshold $\theta \in [-1, 1]$. Our system defines that two faces are matching if and only if their template vectors $\vec{x}$ and $\vec{y}$ satisfy $\cos(\vec{x}, \vec{y}) \geq \theta$.

3 Related Work

3.1 Non-FHE Facial Matching Approaches

Facial recognition is a well-studied topic [17, 18, 63, 64, 69]. Traditional non-FHE approaches for secure facial matching include template-protection and cancelable-biometric schemes that apply non-invertible transforms [25, 60], but accuracy drops and transform specific attacks remain a concern in such methods [41, 54]. Biometric cryptosystems couple templates with cryptographic keys [34] yet are sensitive to intra-user variability. Partial HE schemes (e.g., Paillier) support only limited encrypted operations [62], confining protocols to linear scoring. SMPC and garbled circuits offer general secure computation [20, 59] but incur high communication and complexity for high-dimensional face embeddings.

FHE-based methods for secure-facial matching address critical limitations by enabling arbitrary computations on encrypted data under a single-party trust model. While non-FHE techniques often sacrifice accuracy (e.g., template transformations) or functionality (e.g., restricted HE operations), FHE preserves the native accuracy of facial matching algorithms without exposing raw templates. However, FHE also introduces significant computation overhead and requires careful optimizations to mitigate latency, and challenges less pronounced in SMPC or garbled circuits-based methods.

In the following sections, we present in-depth reviews of state-of-the-art FHE-based methods for facial template matching.

3.2 Baseline FHE Approach

The task of securely matching facial template vectors using homomorphic encryption has been well-studied, and a naive approach to computing template similarities is well-established in the literature. This approach was first outlined by Boddeti *et al.* [5] to handle the 1:1 template matching task, and it utilized the BFV [21] scheme.

Their method utilizes packing (batching) to encrypt multiple values into a single ciphertext, leveraging SIMD properties for efficient element-wise operations. Given a query and a database template vector, both are unit-length normalized and packed into separate ciphertexts. A single multiplication computes element-wise products, followed by $\log_2(\ell)$ rotations and additions to sum the slots, yielding the cosine similarity score in the first slot of the output ciphertext [5]. This process is outlined in Algorithm 1, where $\mathbf{q}$ and $\mathbf{d}$ hold the packed, normalized vectors.

Algorithm 1 Literature Baseline Inner Product Algorithm

```
procedure NAIVE-INNERPRODUCT(d, q)
    Denote output score ciphertext as s
    s ← MULTIPLY(d, q)
    RELINEARIZE(s)
    RESCALE(s)
    for i = 0, 1, ..., log2(ℓ) do
        s ← ADD(s, ROTATE(s, 2^i))
    return s
```

Ibarrando *et al.* [29] extended the baseline approach to the $1 : K$ matching task using CKKS [12] instead of BFV. Given that $N/2 > \ell$ and both are powers of two, their method efficiently packs $N/2\ell$ template vectors into a single ciphertext. By normalizing all vectors and packing the database templates sequentially while replicating the query vector in another ciphertext, cosine similarity scores can be computed in a similar manner as before using Algorithm 1. A key limitation of Algorithm 1 is that similarity scores appear at intervals of ℓ slots, misaligned with database vector positions in $\mathbf{d}_i$, leaving the remaining slots with meaningless values. Therefore, in order to merge the similarity scores into the minimum required number of ciphertexts, $\mathcal{S}$ must use plaintext masking multiplications, additions, and rotations [29]. This score-merge step requires significant computational time and consumes at least one extra level of multiplicative depth. To complete the membership and identification scenarios using the baseline approach, $\mathcal{S}$ first applies Algorithm 1 to compute and merge similarity scores. Then, it homomorphically compares each score to a match threshold, setting values above the threshold to 1 and others to 0. In the membership scenario, $\mathcal{S}$ sums these values into a single ciphertext for C. In the identification scenario, it directly returns the vector of thresholding results.

A major drawback of the baseline score computation algorithm is its reliance on rotation-based summation, which requires relinearization and rescaling after every multiplication [56]. Since each score ciphertext $\mathbf{s}$ holds only $N/2\ell$ similarity scores, $\mathcal{S}$ must perform $\lceil 2K\ell/N \rceil$ relinearization and rescaling operations for a query involving K database vectors. By eliminating rotation-based summation, our protocol reduces this overhead to just $\lceil 2K/N \rceil$ operations. We provide a thorough comparison of the complexity of the baseline with our work in Section E.

3.3 *GROTE*

Ibarrando *et al.* [29] propose *GROTE*, a secure facial matching system that enhances the baseline approach by incorporating homomorphic group testing. Instead of applying costly sign approximation functions to every cosine similarity score, *GROTE* approximates the maximum score within groups and compares only these representative values against the match threshold, reducing computation overhead. Their key contribution was a homomorphic approximation of the maximum element of a vector. Using the α-norm approximation, where $\max(\mathbf{z}) \approx \left(\sum_i |\mathbf{z}_i|^\alpha\right)^{1/\alpha}$ for a large α, they simplified computation by omitting the root, instead approximating $\max(\mathbf{z})^\alpha$ directly. One major drawback of *GROTE* is its restrictive assumption that a query template vector can match at

most one database template vector, which limits its applicability in scenarios where multiple matches may exist [29]. This assumption is not realistic in many real-world use cases, where $\mathcal{S}$'s database could contain multiple images of the same subject. Additionally, *GROTE* inherits the inefficiencies of the baseline approach. By relying on the naive inner product computation from Algorithm 1, it incurs excessive relinearization and requires a costly score-merge operation, leading to higher computation latency for the server.

3.4 *Blind-Match*

Choi *et al.* [16] present *Blind-Match*, an FHE-based 1:K identification protocol. Each template vector is split into subvectors, with each subvector packed into a separate ciphertext; a custom cosine-similarity routine then computes scores using fewer additions and rotations than the baseline approach.

Blind-Match defines an encoding scheme that partitions each database template vector into N_{in} subvectors, each of length ℓ/N_{in}. During enrollment, N_{in} ciphertexts are created, and each subvector is placed at the same position of a different ciphertext. With this encoding scheme, $(N \cdot N_{in})/2\ell$ database template vectors can be packed into each group of N_{in} ciphertexts. To initiate a query, C encrypts the query and transmits it to $\mathcal{S}$. $\mathcal{S}$ then applies plaintext masking multiplications to expand the ciphertext into N_{in} ciphertexts, each containing duplicate copies of the corresponding query. For each group of N_{in} ciphertexts, $\mathcal{S}$ initializes an output score ciphertext $\mathbf{c}_{out}$ and iteratively accumulates the rescaled product of each ciphertext and its corresponding expanded query ciphertext. $\mathcal{S}$ performs $\log_2(\ell/N_{in})$ rotations and additions to aggregate slot values within the subvectors of $\mathbf{c}_{out}$. Consequently, the N_{in} indices store similarity scores, while other indices contain just noise [16]. $\mathcal{S}$ packs the similarity results with a score-merge algorithm and returns the ciphertext to C, who decrypts and interprets the result.

Like prior work, *Blind-Match* relies on black-box multiplications, requiring $\mathcal{S}$ to relinearize and rescale after each multiplication. An algorithm that could eliminate these rotation operations during score computation could significantly reduce relinearization overhead and latency [65]. Blind-Match also relies on N_{in} query ciphertexts instead of a single ciphertext, increasing $\mathcal{S}$'s computation due to a higher multiplicative depth, multiplications, and rotations for query expansion. Finally, *Blind-Match* inherits the baseline's dependence on a score-merge algorithm, necessitating an additional computationally expensive compression step due to its inability to produce similarity scores in sequential order.

3.5 *HERS*

In [19], Engelsma *et al.* introduced *HERS*, a secure facial matching system using the BFV scheme. They introduced a novel database encoding technique that allows for efficient computation of cosine similarity scores in the encrypted domain. From a high-level perspective, the encoding scheme and similarity computation algorithm of *HERS* is analogous to that of *Blind-Match* with $N_{in} = \ell$. *HERS* defines an encoding scheme that packs each dimension of the database template vectors into a separate ciphertext. Therefore, a single template vector would be encoded into ℓ plaintexts with the i-th slot of the vector placed into the first slot of the i-th plaintext. Using SIMD, *HERS* can pack $N/2$ template vectors into each group of ℓ plaintexts, such that the i-th slot of the j-th template vector is placed at the j-th slot of the i-th plaintext [19]. Once all database template vectors have been encoded, each group of ℓ plaintexts can be encrypted and transmitted to $\mathcal{S}$.

Unlike *Blind-Match*, where $\mathcal{S}$ performs query expansion within the encrypted domain, *HERS* requires C to preprocess, encrypt, and transmit all ℓ expanded query ciphertexts. Since query and database vectors are encrypted with their slots aligned across ciphertexts, *HERS* can compute similarity scores, avoiding ciphertext rotation operations. The similarity scores between the $N/2$ database vectors within the ℓ ciphertexts and the query vector can be computed as $\mathbf{s} = \sum_{i=0}^{\ell-1}(\mathbf{q}_i \cdot \mathbf{d}_i)$, where similarity score between the query vector and the i-th database vector is located at the i-th slot of $\mathbf{s}$. This score computation algorithm eliminates many rotation operations and an expensive score-merge operation. While this approach reduces computation overhead on the server, it significantly increases communication overhead for C, as a separate ciphertext must be transmitted for each dimension of the query template (ℓ ciphertexts/query) significantly increasing its end-to-end query latency and making it impractical in bandwidth-constrained environments.

4 Definitions

4.1 System Model

To define the 1:K secure facial matching problem, we first introduce the three involved parties: *enroller* $\mathcal{E}$, *server* $\mathcal{S}$, and *client* C.

$\mathcal{E}$ is a party responsible for setting up the encrypted facial matching database, held by $\mathcal{S}$. To add a subject to the secure facial matching database, $\mathcal{E}$ acquires a photo of their face and uses a facial feature extractor to generate a facial template vector, called a *database template vector*. Once the database template vectors have been collected and preprocessed, $\mathcal{E}$ will encrypt them using the public key and provide the resulting ciphertexts to $\mathcal{S}$ for storage. $\mathcal{E}$ participates only in system setup and plays no role in per-query execution. In prior work, $\mathcal{E}$ is sometimes called the "biometric provider" [29], or merged with C into a single "client" entity [16, 19]. In practice, $\mathcal{E}$ could represent a law enforcement agency uploading facial embeddings of wanted individuals to a secure centralized database.

$\mathcal{S}$ maintains a collection of encrypted facial templates called *database template ciphertexts* that are supplied by $\mathcal{E}$ and homomorphically evaluates all queries over this dataset. In deployment, $\mathcal{S}$ could be a central party or server that stores a list of suspected individuals and their encrypted facial information.

Finally, C is a party possessing a facial image of a subject. C generates a corresponding facial template vector, called the *query template vector*, which they preprocess and encrypt using the public key. The resulting ciphertext is called the *query ciphertext*. C wishes to learn whether their subject matches with any of the subjects stored in the database held by $\mathcal{S}$. We define two different queries C can perform on $\mathcal{S}$'s database: the *Membership scenario* and the *Identification scenario*. In the membership setting, C learns a single bit indicating whether the query vector matches any database template. In the identification setting, C obtains the indices of all matching templates, revealing the associated identities. In a real-world secure facial matching system, C could be a TSA agent at an airport security terminal, submitting a traveler's photo for screening.

4.2 Adversary and Threat Model

We assume a semi-honest threat model where C and $\mathcal{S}$ are both expected to be honest and curious. They execute the protocol without deviation but attempt to learn as much information as possible.

We distinguish the $\mathcal{E}$ from the $\mathcal{S}$ to reflect two deployment realities that arise in practice. First, it allows our protocol to decentralize the enrollment process and centralize storage and computation. It allows field offices or mobile agents (e.g., state Department of Motor Vehicles, individual DHS agents) to capture and preprocess biometric samples, as these parties might lack the infrastructure or resources to host millions of templates or perform large-scale homomorphic evaluations. Second, regulatory frameworks such as the GDPR and CCPA mandate encryption of personally identifiable information (PII) during transmission and storage. Hence, confining encryption to the $\mathcal{E}$ and delegating all subsequent storage and computation to the $\mathcal{S}$ guarantee end-to-end ciphertext protection and avoid redundant PII replication across devices, reducing the attack surface. We therefore model $\mathcal{E}$ and $\mathcal{S}$ as distinct roles, even when they are operated under the same institutional umbrella to capture common real-world deployments and privacy requirements.

In order to satisfy the privacy concerns discussed in the example use cases (see Section 1), we impose significant restrictions on the information accessible to $\mathcal{S}$ and C. All facial template vectors possessed by C and $\mathcal{S}$ must be encrypted using a public key FHE, where only C possesses the private decryption key. With this restriction, we ensure that $\mathcal{S}$ cannot access the PII of C's subject, or the PII of any subjects in $\mathcal{S}$'s database. Since C never has access to the database template vectors, unencrypted or encrypted, we ensure that C can never access the PII of $\mathcal{S}$'s subjects. We also require $\mathcal{S}$ to compute all queries in the encrypted domain, such that only C can access the query results in their unencrypted form. $\mathcal{E}$ is a trusted entity that views the database template vectors in their unencrypted form, and C must be trusted to view its own query template vector in its unencrypted form. We assume there is no collusion between C, $\mathcal{S}$, and $\mathcal{E}$.

In the identification scenario, we require that C cannot learn the exact cosine similarity scores between its query vector and all database vectors. Instead, C is only given access to the indices of the matching database template vectors within the database of $\mathcal{S}$.

With these restrictions, our adversary/threat model closely resembles some previous works [29], and slightly deviates from other previous works [16, 19] since we do not leak exact similarity scores to C and we separate the enroller and client parties. As mentioned before, these protocols require C to perform the responsibilities of $\mathcal{E}$, but such a setting requires C to enroll and, therefore, have plaintext access to every database template vector. Our protocol has a much stronger threat model and does not allow such plaintext access; instead, it entrusts $\mathcal{E}$ for encrypting the database vectors separate from C.

4.3 Protocol and Security Definitions

Definition 1 (Secure Facial Matching Protocol Π). *Let* $\Pi = (\mathsf{Setup}, \mathsf{Enroll}, \mathsf{Query}, \mathsf{Compute}, \mathsf{Extract})$ *be a protocol between three parties: Enroller* $\mathcal{E}$*, Server* $\mathcal{S}$*, and Client* C*.*

- $\mathsf{Setup}(1^\lambda, d) \rightarrow (params, pk, sk, evk)$*: C initializes CKKS parameters for security level* λ *and depth d [3], obtains sk, and distributes pk to* $\mathcal{E}$ *and evk to* $\mathcal{S}$*.*
- $\mathsf{Enroll}(pk, D) \rightarrow \mathbf{M}$*:* Enroll *is run by* $\mathcal{E}$*. It takes as input a database D and outputs the vectors in the correct encrypted format* $\mathbf{M}$*.*
- $\mathsf{Query}(pk, q) \rightarrow \mathbf{q}$*:* Query *is run by C. It takes as input a query vector q, then formats and encrypts it into an encrypted output* $\mathbf{q}$*.*
- $\mathsf{Compute}(evk, \mathbf{M}, \mathbf{q}) \rightarrow \boldsymbol{ct}$*:* Compute *is run by* $\mathcal{S}$*, which outputs the final encrypted result* $\boldsymbol{ct}$*.*
- $\mathsf{Extract}(sk, \boldsymbol{ct}) \rightarrow res$*:* Extract *is executed by C to decrypt and obtain the facial matching results res in plaintext form.*

We consider security against a semi-honest adversary $\mathcal{A}$ controlling $\mathcal{S}$. We assume no collusion between parties, meaning the adversary $\mathcal{A}$ only controls $\mathcal{S}$ and cannot coordinate with other parties to gain additional information. Due to this no-collusion assumption, the adversarial $\mathcal{S}$ never receives decrypted data during the protocol execution. Thus, we need only consider the IND-CPA security model rather than IND-CPA with access to the decryption oracle (IND-CPA^D [44]).

The real view $\mathsf{VIEW}_{\mathsf{real}}$ consists of encrypted protocol values $(\mathbf{M}, \mathbf{q}, \mathbf{ct}, r)$ where $\mathbf{M}$ represents the encyrpted database, $\mathbf{q}$ represents the encrypted query, and $\mathbf{ct}$ respresents the encrypted result. The simulated view $\mathsf{VIEW}_{\mathsf{SIM}}$ of a probabilistic polynomial-time simulator SIM consists of encryptions of randomly sampled values $(\mathsf{SIM}.\mathbf{M}, \mathsf{SIM}.\mathbf{q}, \mathsf{SIM}.\mathbf{ct}, r)$ where $\mathsf{SIM}.\mathbf{M} = \mathsf{Encrypt}(R_D)$ for $R_D \xleftarrow{\$} \mathbb{Z}_q^{l \times d}$, $\mathsf{SIM}.\mathbf{q} = \mathsf{Encrypt}(r_q)$ for $r_q \xleftarrow{\$} \mathbb{Z}_q^d$, and $\mathsf{SIM}.\mathbf{ct} = \mathsf{Encrypt}(r_s)$ for $r_s \xleftarrow{\$} \mathbb{Z}_q^l$.

Definition 2 (Security of Protocol Π). *Protocol* Π *is considered secure against semi-honest adversaries if for every PPT adversary* $\mathcal{A}$ *controlling party* $\mathcal{S}$*, there exists a PPT simulator* SIM *such that for all PPT distinguishers* $\mathcal{D}$*:*

$$|\Pr[\mathcal{D}(\mathsf{VIEW}_{\mathsf{real}}) = 1] - \Pr[\mathcal{D}(\mathsf{VIEW}_{\mathsf{SIM}}) = 1]| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{VIEW}_{\mathsf{real}} = (\mathbf{M}, \mathbf{q}, \boldsymbol{ct}, r)$ *represents the view of server* $\mathcal{S}$ *during a real protocol execution with database D, query q, and random coins r, and* $\mathsf{VIEW}_{\mathsf{SIM}} = \mathsf{SIM}(1^\lambda)$ *represents the simulated view.*

5 Protocol Description: HyDia

Our novel approach to the secure $1 : K$ facial matching task relies on a diagonal representation of database vectors in the encrypted domain, along with hybrid approximations that minimize approximation errors without significant multiplicative depths. The packed diagonal representation enables the extension of an optimized matrix-vector multiplication algorithm to compute K cosine similarity scores with only $\lceil 2K/N \rceil$ relinearization and rescaling operations, compared to $\lceil 2K\ell/N \rceil - \lceil 2K/N \rceil$ in the baseline approach. Our matrix-vector multiplication algorithm also requires a constant $\ell - 1$ rotations, as opposed to over $\lceil 2K\ell/N \rceil \cdot \log_2(\ell)$ in the baseline. A complete complexity comparison with the baseline is given in Table 15 (E). Our computation algorithm also allows for greater parallelism and reduced communication overhead compared to existing state-of-the-art approaches [16, 19]. The hybrid approximations allow homomorphic thresholding with lower errors with comparable multiplicative depths compared to existing

Algorithm 2 Database Vector Diagonal Encoding

```
procedure DIAGONALIZE(M = {M⃗_0, ..., M⃗_{ℓ-1}})
    Denote an unencrypted ℓ × ℓ matrix as D
    for i = 0, ..., ℓ − 1 do
        for j = 0, ..., ℓ − 1 do
            D[i][j] ← M[j][i + j  mod ℓ]
    return D
```

algorithms. In sum, our novel HyDia minimizes end-to-end per-query computation time and communication costs compared to state-of-the-art methods while surpassing existing approximation techniques in both accuracy and efficiency.

The high-level idea for our method begins by considering the task of computing $1 : K$ cosine similarity scores as a singular rectangular matrix-vector multiplication. Given a matrix of normalized database vectors $D \in \mathbb{R}^{K \times \ell}$ and a normalized query vector $\overline{q} \in \mathbb{R}^{\ell}$, we can compute the scores $\vec{s} \in \mathbb{R}^{K}$ between the query and database vectors as $\vec{s} = D\overline{q}$. Halevi and Shoup [26] pointed out that, by diagonalizing a square matrix, a square matrix-vector multiplication can be computed efficiently using the element-wise addition, multiplication, and rotation operations supported by FHE schemes.

However, in many real-world $1 : K$ facial matching scenarios, K will be substantially larger than ℓ, requiring cosine similarity scores to be computed via rectangular matrix-vector multiplication rather than a square one. Therefore, we extend Halevi and Shoup's diagonal multiplication algorithm to the $1 : K$ facial matching task by developing an encoding technique which efficiently packs many square diagonalized matrices of template vectors into ciphertexts, allowing HyDia to efficiently perform rectangular matrix-vector multiplication using Halevi and Shoup's optimized algorithm. As we will show, our packed diagonal encoding will allow us to eliminate intermediate rotation operations during our score computation algorithm, just as the stacked encoding of *HERS* eliminates intermediate rotations. Crucially, our packed diagonal encoding will only require a single ciphertext to be generated by the client per query, as opposed to ℓ ciphertexts in *HERS*, greatly reducing client overhead and communication costs.

5.1 Database Encryption with Diagonalization

Similar to other works utilizing novel encodings [16, 19], our protocol begins with $\mathcal{E}$ preprocessing and encrypting the database template vectors as a one-time step.

During this diagonal encoding step, $\mathcal{E}$ considers ℓ template vectors at a time, denoted as a matrix $M = \{\vec{M}_0, ..., \vec{M}_{\ell-1}\}$ (if fewer than ℓ vectors remain, we append dummy-zero-value vectors to M). More formally, Algorithm 2 defines how individual slots of M are encoded into the diagonal matrix D. Note that a mod operation is applied to the column indices so that each diagonal vector wraps around the columns of the matrix M. $\mathcal{E}$ defines an $\ell \times \ell$ matrix M such that the i-th row of M stores the i-th diagonal vector of D. Figure 2 illustrates an example of this diagonal encoding for $\ell = 4$, with the rows of M being database template vectors, and the rows of D being diagonal vectors of M.

In order to perform the similarity score computation algorithm, each row of D must be encrypted into a separate ciphertext, requiring ℓ mostly-unfilled ciphertexts. However, in the relevant case

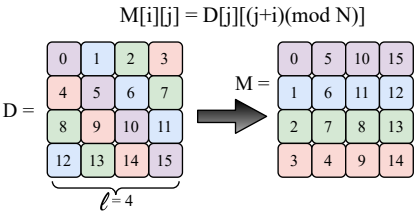

Figure 2: Example of diagonal matrix encoding

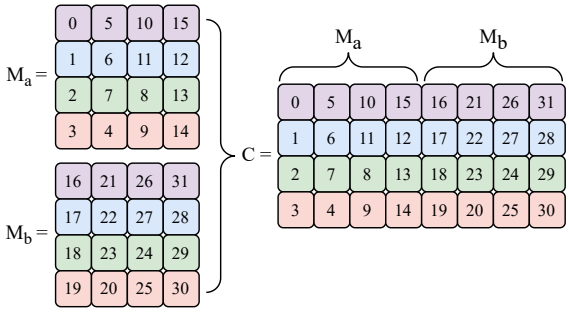

Figure 3: Example of side-by-side matrix packing via SIMD

where K is larger than ℓ, we split the template vectors into $\lceil K/\ell \rceil$ matrices, then diagonalize each into $\{M_0, M_1, ...\}$ before $\mathcal{E}$ can encode the template vectors. The rows of diagonal matrices can be packed in a side-by-side manner in each of the ℓ ciphertexts. More specifically, the i-th ciphertext in this scenario will include the i-th row of M_0, followed by the i-th row of M_1, until the ciphertext has been completely filled. Figure 3 illustrates how multiple diagonalized matrices can be packed side-by-side, such that every row of the resulting concatenation C is a plaintext that can then be encrypted into a singular ciphertext.

5.2 Single-Ciphertext Query Generation

To begin an individual query, C preprocesses and encrypts the template vector corresponding to the query subject. In our protocol, C completes this step exactly as they would in the FHE baseline approach (Section 3.2). Given a template vector of length ℓ, C first unit-length normalizes the vector in the unencrypted domain. Then, C packs a plaintext with $(N/2)/\ell$ copies of the query vector in a sequential manner, where $N/2$ is the number of elements that can be encrypted/packed into a single ciphertext via SIMD capabilities of the CKKS scheme. C then encrypts this packed plaintext using the public key of the system, and transmits the single resulting ciphertext to $\mathcal{S}$ as their query. A key advantage of HyDia is that it requires only a single ciphertext to be encrypted and transmitted per query, whereas other state-of-the-art methods [16, 19] require multiple (up to 512) ciphertexts for a single query.

5.3 Optimized Similarity Computation

During the setup of the system, we ensure that C provides $\mathcal{S}$ with the generated multiplication and rotation keys. Therefore, once C has provided $\mathcal{S}$ with the encrypted query vector as described above, $\mathcal{S}$ can proceed to compute the cosine similarity scores between the query vector and each database template. $\mathcal{S}$ reuses the same multiplication and rotation keys for each query C provides; C does not need to transmit its keys more than once.

Algorithm 3 Diagonal Similarity Score Computation

```
procedure DIAGONAL-SCORES(D, Q)
    Denote vector of ℓ ciphertexts as S
    for i = 0, ..., ℓ − 1 do                    ▷ Loop run in parallel
        S_i ← MULTIPLY(D_i, Q_i)
    for i = 1, ..., ℓ − 1 do
        S_0 ← ADD(S_0, S_i)
    RELINEARIZE(S_0)
    RESCALE(S_0)
    return S_0
```

Algorithm 4 Secure Thresholding with Hybrid Approximation

```
procedure HOMOMORPHIC-THRESHOLD(x, θ, κ)    ▷ x is an encrypted score
    x' ← ChebyshevApprox(x, κ − 4)          ▷ Approximate c'(x)
    x'' ← f_4(x')                           ▷ Refine boundary errors via f_4
    ct ← ADD(x'', 1)                        ▷ Shift results along y-axis
    return ct
```

$\mathcal{S}$ begins the similarity computation by performing and storing all ℓ distinct rotations of the query ciphertext. In Halevi and Shoup's diagonal transform algorithm, the transformed vector needs to be rotated ℓ times, such that each rotation can be used in a separate product [26]. However, in the case where the database possesses many batches of ℓ ciphertexts (i.e. $K > N/2$), we would need to apply this algorithm several times. To prevent $\mathcal{S}$ from repeating the same rotations of the query ciphertext, we compute and store all ℓ distinct rotations immediately. While performing $\ell - 1$ rotation operations would typically be computationally expensive, we can leverage the "hoisting" optimization [27] and also perform the rotations in parallel to make this step efficient. Since we wish to rotate *the same* vector by multiple different rotation factors, the hoisting optimization allows us to precompute portions of the rotation operation which would be identical for all different factors [33]. Therefore, we only need to perform those expensive portions once as opposed to $\ell - 1$ times. By using hoisted rotations, we are able to reduce the server overhead by a constant amount, averaging over 6 seconds in both scenarios, with Table 11 in Section C displaying the full comparison. We provide a more detailed explanation on the working of hoisting optimization in Section D. In addition to hoisting, we can perform all $\ell - 1$ rotations in parallel, with each thread being able to use the same common precomputed values.

$\mathcal{S}$ now possesses a vector of ciphertexts Q, where each Q_i is the query ciphertext rotated left by i. $\mathcal{S}$ then performs Algorithm 3 to compute similarity scores between the query ciphertext and each batch of ℓ database ciphertexts possessed by $\mathcal{S}$. Algorithm 3 is an FHE-based implementation of Halevi and Shoup's diagonal matrix-vector multiplication algorithm introduced earlier, extended in this work to handle rectangular matrices. We provide a correctness proof of this extended algorithm in Section A. The result of Algorithm 3 is a ciphertext containing the encrypted similarity scores between the query template and the database templates encrypted within that specific batch.

One of our most important novel optimizations is not to use black-box multiplications in our score computation algorithm, unlike previous methods. Instead, our protocol multiplies the query and database ciphertexts without immediately relinearizing or rescaling them. Since we can add three-element ciphertexts with larger scaling factors, we delay those operations until all ℓ product ciphertexts have been summed into a single output score ciphertext. Therefore, for a query involving K database template vectors, our score computation algorithm requires only $\lceil 2K/N \rceil$ relinearization and rescaling operations.

Our approach efficiently leverages parallel computing during the score computation step, as all ℓ ciphertext multiplications can be done independently. Since these multiplications are the most time-consuming operations in our protocol, this parallelization significantly reduces per-query computation time.

5.4 Hybrid Approximation for Homomorphic Thresholding

After cosine similarity scores are computed homomorphically, $\mathcal{S}$ homomorphically compares them to a threshold $\theta \in [-1, 1]$ to generate encrypted matching results. θ is a constant and a public parameter that has been agreed upon by all parties during the setup phase of the protocol.

The ideal behavior of the threshold function is that of a step function that maps values below θ to 0 and values equal to or above θ to a constant. This way, in the membership scenario, the sum of all compared scores will equal 0 if and only if there are no matches between the query and database vectors. In the identification scenario, applying this function to the encrypted scores prevents C from learning the exact similarity scores, preventing potential reconstruction attacks (Section 1). Since directly computing a step function is not supported by CKKS, it must be approximated using some polynomial approximation techniques.

Chebyshev Approximation. The Chebyshev polynomial approximation method is a minimax-based polynomial method that approximates any (nonlinear) function with low polynomial degrees and minimal approximation error [14, 35], and it is a state-of-the-art approximation for FHE [28, 40, 42, 43, 58]. Since the domain of cosine similarity scores falls under [-1, 1], our approximation of the threshold function only needs to consider that domain.

We observed that while the Chebyshev approximation method achieves low error near the discontinuity at $x = \theta$, it introduces non-negligible errors near the domain boundaries $[-1, 1]$. In the membership scenario with many database vectors, adding together many non-matching scores with such errors would accumulate and result in a value greater than θ, which would lead to a false positive. To mitigate this, we designed a hybrid thresholding function that refines the Chebyshev approximation by applying an additional function with minimal errors near the domain boundaries $[-1, 1]$.

Cheon et al. Approximation. Another suitable approximation for a step function, given by Cheon *et al.* [13], is $f_4(x) = \frac{35}{128}x^9 - \frac{180}{128}x^7 + \frac{378}{128}x^5 - \frac{420}{128}x^3 + \frac{315}{128}x$, defined over $[-1, 1]$. $f_4(x)$ is extremely close to 1 when x is close to 1 and extremely close to -1 when x is near -1. As described in [13], f_4 can be composed with itself several times to achieve an approximation of the sign function. However, several such compositions will incur a large level of multiplicative depth for an inaccurate sign approximation, compared to the Chebyshev approximation method. Despite this, a single application of f_4 can

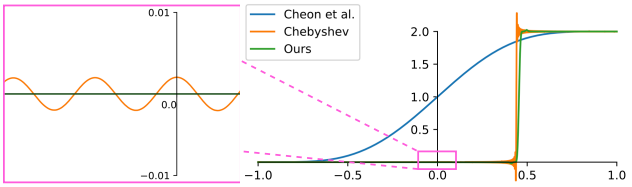

Figure 4: Comparison of our hybrid approximation against Chebyshev ($\theta = 0.44$, $\kappa = 10$) and Cheon *et al.*'s $f_4 + 1$. Note that ours overlaps with the x-axis in the zoomed figure.

be useful in minimizing the errors of the Chebyshev approximation at the edges of [-1,1], at the acceptable cost of four levels of multiplicative depth and an overhead of 0.389s.

Novel Hybrid Approximation. Our novel idea is to compose the Chebyshev approximation with $f_4(x)$. We use the Chebyshev method to approximate the step function $c'(x)$, which is defined to equal -1 when $-1 \leq x < \theta$ and to equal 1 when $\theta \leq x \leq 1$. We approximate $c'(x)$ specifically so that the range of the resulting values aligns with the domain of $f_4(x)$, being [-1, 1]. Instead of allotting κ levels of multiplicative depth to the Chebyshev polynomial approximation, our hybrid approach will instead allot $\kappa - 4$ level to the Chebyshev approximation and the remaining 4 levels to the computation of f_4. Therefore, we use a Chebyshev polynomial with $\kappa - 4$ levels of depth to approximate $c'(c)$, then pass those results through $f_4(x)$. Finally, we adjust the resulting value by adding 1, ensuring that non-matching scores are mapped to 0 and the matching ones are mapped to 2. Algorithm 4 shows the algorithm.

Figure 4 shows our hybrid approximation technique evaluated over the domain $[-1, 1]$ with a threshold $\theta = 0.44$ and depth $\kappa = 10$ (θ is optimized with experiments in Section 7.1). Our hybrid technique (green curve) clearly maintains a lower approximation error than the pure Chebyshev method (orange) across most of the domain, as highlighted in the zoomed region on the left. While a small error remains near $x = \theta$ due to the inherent challenge of approximating the step's discontinuity, our approach overall stays close to the ideal step function for values away from θ.

Moreover, our hybrid approach offers better computation overhead compared to the Chebyshev polynomial approximation at the same depth. The highest-degree Chebyshev polynomial that can be computed using $\kappa = 10$ levels of multiplicative depth is a 1007-degree polynomial, which incurs a 3.79s overhead. In contrast, our hybrid approximation uses a 59-degree Chebyshev polynomial consuming 6 levels of multiplicative depth, followed by f_4 consuming 4 levels of depth, for a combined overhead of 1.19s. Our hybrid approach not only reduces the errors of the Chebyshev approximation polynomial away from the discontinuity but also incurs significantly lower computation overhead than the most accurate Chebyshev polynomial at the same depth.

5.5 The Complete HyDia Protocol

By integrating all aforementioned optimizations, we present the complete protocol of HyDia for the $1 : K$ matching task for both membership and identification scenarios (Figure 5).

If C wishes to perform the membership scenario with their encrypted query ciphertext, then $\mathcal{S}$ performs Algorithm 5 through the membership scenario branch, whereupon they sum all comparison ciphertexts into $\mathbf{ct}_0$, sum all the slots of $\mathbf{ct}_0$, then return

Algorithm 5 Server Query Computation

procedure COMPUTE-QUERY($\mathbf{M}, \mathbf{q}$)
  Denote vector of ℓ rotated query ciphertexts as $\mathbf{Q}$
  for $i = 0, ..., \ell - 1$ **do** ▷ Parallel
    $\mathbf{Q}_i \leftarrow$ ROTATE($\mathbf{q}, i$) ▷ Using hoisted rotations
  Denote vector of $\lceil 2K/N \rceil$ score ciphertexts as $\mathbf{S}$
  Denote vector of $\lceil 2K/N \rceil$ comparison ciphertexts as $\mathbf{ct}$
  for each group of ℓ ciphertexts $\mathbf{M}_i \in \mathbf{M}$ **do** ▷ Parallel
    $\mathbf{S}_i \leftarrow$ DIAGONAL-SCORES($\mathbf{M}_i, \mathbf{Q}$)
    $\mathbf{ct}_i \leftarrow$ HOMOMORPHIC-THRESHOLD($\mathbf{S}_i, \theta, \kappa$)
  if handling Identification scenario **then**
    return ct
  else if handling Membership scenario **then**
    for $i = 1, ..., (2K/N) - 1$ **do**
      $\mathbf{ct}_0 \leftarrow$ ADD($\mathbf{ct}_0, \mathbf{ct}_i$)
    for $i = 0, ..., \log_2(N/2) - 1$ **do**
      $\mathbf{temp} \leftarrow$ ROTATE($\mathbf{ct}_0, i$)
      $\mathbf{ct}_0 \leftarrow$ ADD($\mathbf{ct}_0, \mathbf{temp}$)
    return $\mathbf{ct}_0$

that single ciphertext to C. After obtaining this ciphertext, C can decrypt it using the secret key sk of the FHE scheme. Every slot of this ciphertext will contain the same value, with a value less than 1 indicating that no facial matches were found between the query template vector and all of the database template vectors. A value greater than or equal to 1 indicates to C that at least one facial match exists between their query and $\mathcal{S}$'s database.

If C wishes to perform the identification with their query ciphertext, then $\mathcal{S}$ performs Algorithm 5 through the identification scenario branch, whereupon they return the entire vector of comparison ciphertexts $\mathbf{ct}$ to C. C can then decrypt each ciphertext in the vector using the secret key and examine the values at each index. Similar to the membership scenario, a value <1 at the i-th index indicates no facial match between the query template vector and the i-th database template vector. Furthermore, a value greater than or equal to 1 at the i-th index indicates a facial match between the query template vector and the i-th database template vector.

5.6 Correctness and Security of HyDia

THEOREM 1 (CORRECTNESS OF HYDIA). *Let $D = d_1, d_2, ..., d_k$ be a database of K facial template vectors, each of dimension ℓ, and q be a query template vector of the same dimension. Let θ be the designated matching cosine similarity threshold. Then HyDia correctly achieves the following in two different scenarios:*

(1) Membership scenario: Returns a value > 0 if and only if there exists at least one vector $d_i \in D$ such that $\cos(d_i, q) \geq \theta$.
(2) Identification scenario: For each index i, returns a value $\geq$ if and only if $\cos(d_i, q) \geq \theta$.

To prove the correctness of HyDia, we show that it accurately identifies facial matches based on cosine similarity. The proof follows the flow of the protocol's execution. First, we establish that normalizing both database and query vectors ensures their dot products directly represent cosine similarities. Next, we apply Lemma 2 (Section A) to show that the diagonalization process and subsequent homomorphic operations correctly compute these dot products despite the encrypted domain constraints. For the aggregation phase,

Parameters: ℓ is the dimension of each facial template vector. We assume ℓ is a power of 2. D is the ordered set of database template vectors of size K and $\vec{q}$ is the query template vector initially held by $\mathcal{E}$ and C respectively. θ and κ are the matching cosine similarity threshold and multiplicative depth allotted to the approximation function. $\kappa + 1$, N, and λ are the total multiplicative depth, ring dimension, and the computational security parameter of the CKKS scheme, respectively.

1. **System Setup**
   a. **System parameters**: Parties agree upon parameters (θ, c) for the facial matching system and (N, λ) for the CKKS scheme.
   b. **Key generation**: C executes the SETUP and KEYGEN CKKS algorithms, obtaining the public key pk, the private key sk, and the evaluation key evk. This evaluation key is comprised of the multiplication key and the rotation keys. C then provides pk to $\mathcal{E}$ and evk to $\mathcal{S}$.
2. **Database Enrollment**
   a. **Normalization**: For every database vector $d_i \in D$, $\mathcal{E}$ computes its unit-length normalization $\overline{d}_i = (d_i/||d_i||)$.
   b. **Diagonalization**: $\mathcal{E}$ partitions the ordered database templates into groups of ℓ, forming $\lceil 2K/N \rceil$ matrices, each denoted as $Matrix_i$. If ℓ does not evenly divide K, zero-filled dummy vectors are appended. Each matrix is then diagonalized as $Diag_i = \text{DIAGONALIZE}(Matrix_i)$.
   c. **Matrix packing**: For every $N/2\ell$ diagonalized matrices, $\mathcal{E}$ will define a group of ℓ plaintexts. $\mathcal{S}$ then packs the first row of each of the $N/2\ell$ diagonalized matrices sequentially into the first plaintext. $\mathcal{E}$ repeats this step for all ℓ rows. If there are more than $N/2\ell$ diagonalized matrices, $\mathcal{E}$ defines another group of ℓ plaintexts and repeats the packing process.
   d. **Encryption**: Using pk, $\mathcal{E}$ encrypts plaintexts into $\lceil 2K/N \rceil$ groups of ℓ ciphertexts, denoted as $\mathbf{M}_i \in \mathbf{M}$, and provides $\mathbf{M}$ to $\mathcal{S}$.
3. **Client Query Generation**
   a. **Normalization**: C computes the unit-length normalization $\overline{q} = (\vec{q}/||\vec{q}||)$ of the query template vector.
   b. **Plaintext packing**: C fully packs a single plaintext up to the batchsize $N/2$ with the normalized query vector $\overline{q}$ in sequential order.
   c. **Encryption**: Using pk, C encrypts its plaintext into the query ciphertext $\mathbf{q}$ and provides it to $\mathcal{S}$.
4. **Server Query Computation**
   a. **Query ciphertext expansion**: Using hoisted rotations, $\mathcal{S}$ rotates the query ciphertext by all factors from 0 to $\ell - 1$, storing the results in a vector of ciphertexts denoted $\mathbf{Q}$. All rotations can be performed in parallel.
   b. **Similarity score computation**: For each group of ℓ database ciphertexts, denoted as $\mathbf{M}_i$, $\mathcal{S}$ computes $\mathbf{S}_i = \text{DIAGONAL-SCORES}(\mathbf{M}_i, \mathbf{Q})$
   c. **Threshold comparison**: For each score ciphertext $\mathbf{S}_i$, $\mathcal{S}$ computes the hybrid approximation function using κ levels of multiplicative depth to obtain the comparison ciphertext $\mathbf{ct}_i = \text{HOMOMORPHIC-THRESHOLD}(\mathbf{S}_i, \theta, \kappa)$.
   d. **Query result computation**:
      i. **For Membership scenario**: $\mathcal{S}$ homomorphically adds all other comparison ciphertexts to the first ciphertext $\mathbf{ct}_0$. Then, $\mathcal{S}$ uses $\log_2(\ell)$ rotations and additions to summate all the slots of $\mathbf{ct}_0$. $\mathcal{S}$ returns $\mathbf{ct}_0$ to C as the query result.
      ii. **For Identification scenario**: $\mathcal{S}$ returns the entire vector of comparison ciphertexts $\mathbf{ct}$ to C as the query result.
5. **Client Result Extraction**
   a. **Decryption**:
      i. **For Membership scenario**: Using sk, C decrypts the query result ciphertext into a result vector.
      ii. **For Identification scenario**: Using sk, C decrypts all ciphertexts in the vector of ciphertexts they obtained from $\mathcal{S}$. C packs all of the plaintext values into a single result vector, preserving their order.
   b. **Conclusion**:
      i. **Membership scenario**: The result vector contains identical values across all slots. A value of $v = 0$ indicates the C about no match between the query template $\vec{q}$ and any database template $d_i \in D$, while a value of $v > 0$ confirms at least one match.
      ii. **Identification scenario**: C iterates over the result vector, where each value corresponds to a database template. A value $v_i \geq 1$ at index i indicates a match with the i-th database template.

Figure 5: Complete HyDia query protocol for Membership and Identification scenarios

we verify that our rotation and summation operations preserve the comparison results while efficiently formatting them for client interpretation. Finally, we show that after decryption, HyDia produces positive values for matches in the Membership scenario, and values $v_i \geq 1$ at corresponding indices in the Identification scenario. We defer a more detailed formal proof to Section A.

THEOREM 2 (SECURITY OF HYDIA PROTOCOL). *Assuming CKKS is IND-CPA secure, HyDia is secure against semi-honest adversaries with security parameter λ.*

Again, we provide an informal proof sketch here and defer our formal proof of the protocol's security to Section B.

To prove Theorem 2, we construct a simulator that can generate a view indistinguishable from the real protocol execution, demonstrating that a semi-honest server learns nothing beyond what is intended. We establish security through a hybrid argument with four distributions: H_0: the real-world protocol execution, H_1: first hybrid with simulated database matrix, H_2: second hybrid with simulated query vector, and H_3: complete simulation (our target). We can gradually replace each real protocol component with randomly generated encrypted values without losing the indistinguishability due to the IND-CPA of the CKKS scheme. For each transition between hybrids (H_0 to H_1, H_1 to H_2, H_2 to H_3), we prove that any distinguisher with non-negligible advantage would break the CKKS scheme. Finally, by applying the triangle inequality, we show that the total distinguishing advantage between the real protocol (H_0) and the simulation (H_3) is negligible, completing our proof that the protocol reveals nothing beyond the intended computation results.

6 Discussions

6.1 Multi-Client Considerations

To accommodate multiple clients C_i querying an encrypted database, HyDia can adopt a (t, n)-threshold FHE [6]. We note that employing threshold FHE requires at least $n \geq 3$ non-colluding servers $\mathcal{S}_j$ that collectively hold shares of the decryption key. One of the servers would store the encrypted database, perform all homomorphic evaluations, and provide the encrypted result to the decrypting servers. Any $t > \lfloor n/2 \rfloor$ shares from the servers holding decryption shares (including the computing server) would then suffice for the clients to complete the decryption by combining them locally. Such additions make the protocol susceptible to IND-CPA$^{\text{D}}$ attack, which has been shown to be mitigated using noise-flooding techniques [11]. Alternatively, one could also use proxy re-encryption (PRE)

[52, 57] to support a multi-client setting. Under PRE, $\mathcal{S}$ stores a distinct re-encryption key rk_{e_c} for every client public key pk_c. After computing the score under the key pk_e of $\mathcal{E}$, $\mathcal{S}$ applies rk_{e_c} to convert that ciphertext into one decryptable by pk_c, and then returns the re-encrypted result to the client.

Threshold-FHE performs a one-time distributed key-generation that gives each decrypting server a secret-key share and publishes a common public key; re-keying is needed only when server membership changes, eliminating any single point of compromise but adding t partial decryptions per query and assuming an honest, available server majority. PRE, in contrast, stores a distinct re encryption key rk_{e_c} for every client public key at the computing server, avoiding partial-decryption overhead yet imposing key-management that grows linearly with the client set and requiring clients to fully trust $\mathcal{S}$ to protect those keys and re-encrypt results correctly. We leave such extensions for multi-client support to future work.

6.2 Reducing False Positives

Approximating the threshold function with a Chebyshev polynomial introduces a small positive bias such that encrypted similarity scores strictly below θ are mapped to $\epsilon > 0$ rather than 0. In a membership query over a large corpus containing no true matches, these ϵ-values can accumulate and yield a non-zero aggregate, producing a false positive that the plaintext protocol would avoid. This risk can be reduced by increasing the FHE depth κ, thereby using a higher degree approximation with lower residual error. However, a larger κ increases the CKKS depth and the server's computation overhead. Hence, practical deployments must trade off the probability of such false positives against the added computational cost.

6.3 Increasing Throughput

We can increase the throughput of our protocol in two ways. First, the ciphertext slot count can be increased by increasing the FHE ring dimension, which can pack more vectors per ciphertext. However, it will increase the ciphertext size and force higher FHE parameters to preserve the target security level, thereby increasing communication and computation costs. One can also reduce the feature-vector dimension below 512 for templates, lowering the per-query workload. However, this risks accuracy loss and deviates from the 512-D convention established by current feature extractors [17]. We therefore retain 512-D embeddings and leave a systematic dimension-accuracy study to future work.

7 Experiments

We implemented HyDia and all other approaches (literature baseline [5, 29], *GROTE* [29], *Blind-Match* [16], and *HERS* [19]) using C++17 and OpenFHE v1.2.3 [4]. The source code is available at https://github.com/n7koirala/image_matching/. We used the CKKS scheme [12] for each approach and employed the default FHE parameters provided in OpenFHE [2], which give 128-bit security. Our experiments were run on a server with an Intel Xeon Gold 5412U processor (128GB RAM, 48 logical cores) running Ubuntu 20.04. Due to space limitations, we only present the most relevant results. The complete set of results is provided in Section C. In all applicable tables, the best or most optimal results are highlighted in **bold**.

Table 2: Accuracy Testing Results

Protocol	True Positive	False Negative	True Negative	False Positive	Precision	Recall	F_1-Score
InsightFace	6,921	42	2,204,435	2	0.9997	**0.9940**	**0.9968**
HyDia	6,920	43	2,204,436	1	**0.9999**	0.9938	**0.9968**

7.1 Accuracy Experiments

Our accuracy experiments were performed over a subset of the FRGC 2.0 RGB dataset [55]. Our testing subset consists of 44,278 RGB images from 568 identities. To obtain 512-dimensional feature embeddings, we use the publicly available pre-trained InsightFace model [30], which is based on the ResNet-50 architecture. This model is trained on the WebFace dataset [72] and employs the ArcFace loss function [17]. To evaluate our protocol's accuracy, we partitioned the face dataset into 50 query subjects and 44,228 database subjects, simulating 50 unique identification queries. Among the 2,211,400 total query-database pairs, 6,963 were matching identities (positives) and 2,204,437 were distinct identities (negatives).

To determine the optimal match parameter θ, we computed cosine similarity scores between InsightFace feature embeddings for all query-database pairs in the plaintext domain. The score distributions revealed a clear separation between matching and distinct pairs, allowing us to systematically evaluate precision and recall across match thresholds ranging from 0 to 1 in 0.01 increments (Figure 7 in Section C). To balance precision and recall, we selected the match threshold that maximized the unencrypted model's F_1-score. This approach yielded an optimal threshold of $\theta = 0.44$. Across 50 unique identification scenario queries using this threshold, the plain InsightFace model produced 2 false positives and 42 false negatives, for an F_1-score of 0.9968, and HyDia applied on top of InsightFace produced 1 false positive and 43 false negatives, maintaining the same F_1-score of 0.9968. Table 2 details our accuracy results. Our results demonstrate that our homomorphic thresholding has negligible influence on the face matching accuracies. We could not find larger datasets of real human subjects for larger-scale experiments, but our hybrid approximation yields extremely high fidelity (Figure 4), so we hypothesize HyDia will have negligible accuracy loss in larger-scale scenarios as well.

We focus on evaluating the accuracy of HyDia rather than comparing accuracies across all other literature approaches for two key reasons. First, baseline, Blind-Match, and HERS do not employ any homomorphic thresholding method; instead, they leak the exact similarity scores to the client in the identification scenario. Consequently, these approaches do not suffer the accuracy impact of an approximation scheme, aside from minimal quantization/encoding errors inherent to the CKKS scheme itself. Second, while GROTE introduces a homomorphic comparison, it requires the facial datasets at the server to have at most one match per query, making it incompatible with our multi-match real-world face dataset. Therefore, we omit accuracy comparisons for these approaches, as their designs do not rely on secure score comparisons that could affect performance or accuracy.

7.2 Scalability Experiments

Our scalability experiments were performed over large-scale synthetic datasets, with random noise vectors of dimension $\ell = 512$ serving as the template vectors. The number of database template vectors ranged from 2^{10} to 2^{20}.

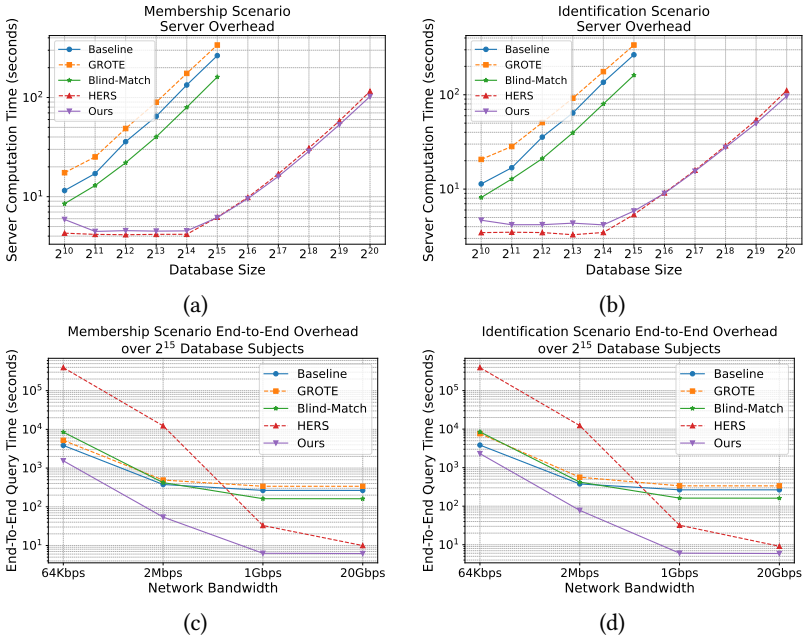

Figure 6: (a, b) Server overhead across approaches by database size; (c, d) End-to-End overhead on various network bandwidths. ** Figures are drawn on a log scale. Tables 6 and 7 in Section C present the exact numbers data behind these charts.

Table 3: Parameters for Each Protocol's Approach

Approach	N	End-to-end Depth	Ciphertext Size	Ciphertexts Sent per Membership	Ciphertexts Sent per Identification
Baseline	2^{16}	13	14.00MB	2	2
GROTE	2^{16}	18	19.00MB	2	3
Blind-Match	2^{16}	12	13.00MB	$N_{in}+1$	$N_{in}+1$
HERS	2^{15}	11	6.00MB	$\ell+1$	$\ell+2$
HyDia (Ours)	2^{15}	11	6.00MB	2	3

We conducted scalability experiments on our implemented protocol by performing ten membership scenario queries and ten identification scenario queries per dataset, and recording the average computational time consumed by the client and the server in each scenario. We also recorded the number of ciphertexts communicated between the client and the server. We applied the same procedure to implement and evaluate the four state-of-the-art protocols from the literature. We note that we could only evaluate the baseline, *GROTE*, and *Blind-Match* on datasets up to 2^{15} vectors, as their excessive memory usage exceeded the available memory in our system. Consequently, for these approaches, we limited our data collection to the six smallest dataset sizes. For each execution of the protocols, we ensured $\lambda = 128$-bit computational security, as well as protocol parameters $\theta = 0.44$ and $\kappa = 10$. Table 3 includes all relevant parameters used in each execution.

7.2.1 Client Computation Overhead. All approaches except *HERS* incur negligible client overhead (less than 0.2s) for generating and encrypting query ciphertext(s). *HERS* requires the client to encode and encrypt $\ell = 512$ ciphertexts in each query, resulting in 2.54s of overhead. Raw data are available in Table 5 in Section C.

7.2.2 Server Computation Overhead. We evaluate baseline, *GROTE*, *Blind-Match*, *HERS*, and HyDia by executing 10 trial runs per database size on a 48-core system and taking the average. Figure 6(a) and Figure 6(b) present the average computation time during the membership scenario and the identification scenario.

HyDia is faster than all four state-of-the-art approaches, demonstrating significant performance improvements over the baseline, *GROTE*, and *Blind-Match* approaches in both the membership and identification scenarios across all database sizes. In comparison to the *HERS*, the prior fastest approach, HyDia showed an improvement of almost 15%: HyDia's average membership computation time is 102.41s and average identification computation time is 96.52s for the 2^{20} database, while the *HERS* approach resulted in average times of 115.54s and 110.91s respectively (full data available in Table 6 in Section C). *HERS* significantly reduces the server's overhead by requiring clients to generate and transmit ℓ ciphertexts ($\ell = 512$ for face template vectors of 512 dimensions) for each query. Even though HyDia has a slightly better computation overhead compared to the prior fastest approach *HERS*, its query contains one ciphertext only, resulting in significant benefits in end-to-end overhead, including communication overhead.

7.2.3 Parallelism Experiments. We also evaluate the combined client and server overheads of each approach for various numbers of maximum cores allotted, from 2 to 48, each averaged over three trials. We display the server-side runtimes for a 2^{15}-subject database in Figure 8 and Figure 9 of Section C, and report the full results in Table 12. HyDia has the lowest latency at every point: membership (identification) latency falls from 26.8s (24.9s) on 2 cores to 6.6s (6.1s) on 48 cores, a 4.1× speed-up. All others either show marginal speedups or increased run time (due to the cost of thread management). Baseline and GROTE achieve similar relative scaling (≈4.4–4.5×) but remain 1–2 orders of magnitude slower, while Blind-Match gains virtually no benefit and sometimes regresses. HERS enjoys relatively higher speed-up (≈6.5 ×) yet still

lags behind HyDia by 60–150%. Thus, HyDia uniquely combines the lowest absolute latency with solid multicore scalability; we expect additional gains beyond 48 cores because its dominant kernel evaluates 512 fully parallel homomorphic multiplications per batch.

7.2.4 End-to-End Overhead including Communication. We define the end-to-end overhead of a facial matching system to be the sum of the client computation, sender computation, and total communication overhead. It is the total time it takes for the client to obtain the decrypted results after initiating the protocol. For a given protocol, the total communication overhead is the time it takes for the client and server to transmit all necessary ciphertexts over the network. Since we performed our experiments on a single machine, we recorded the amount and sizes of ciphertexts that needed to be transmitted, allowing us to compute the total communication overhead for a given network bandwidth.

Using the parameters in Table 3, we can calculate the total transmission size of different protocols for each scenario when querying a database with 2^{15} encrypted templates. We used a 2^{15} size because the first three approaches do not scale to larger databases. The baseline approach has a total transmission size of 28.01MB for both the membership and identification scenarios, whereas *GROTE*'s transmission sizes are 38.01MB and 57.01MB, respectively. Using $N_{in} = 4$, as identified by Choi *et al.* as the optimal number of subvectors, *Blind-Match* produced an overhead of 65.02MB in both scenarios. For $\ell = 512$, as utilized in all our experiments, *HERS* incurs transmission overheads of 3079.45 MB (Membership) and 3085.48 MB (Identification). With 2Mbps/1Gbps bandwidths, *HERS* incurs 12317.9s/24.1s (Membership scenario) and 12341.9s/24.1s (Identification scenario) communication overhead. HyDia achieves significant improvement over all other approaches by incurring transmission sizes of only 12.01MB in the membership scenario and 18.01MB in the identification scenario because HyDia's query contains only one ciphertext and it has lower multiplicative depths, resulting in smaller ciphertexts. With 2Mbps/1Gbps bandwidths, HyDia incurs only 48s/0.10s (Membership scenario) and 72s/0.10s (Identification scenario) communication overhead. Figure 6(c) and Figure 6(d) show the end-to-end overhead of all five approaches across different possible network bandwidths, when querying an encrypted database with 2^{15} template vectors. The complete data on communication/end-to-end overhead are provided in Section C.

7.3 Memory Cost Experiments

7.3.1 Server Disk Costs. In order for $\mathcal{S}$ to hold K encrypted database templates, as prescribed by HyDia, $\mathcal{S}$ must store $\ell \cdot \lceil 2K/N \rceil$ ciphertexts. Using the experimental parameters for HyDia in Table 3, we can calculate the total memory overhead required for $\mathcal{S}$ as approximately $3 \cdot \lceil K/2^{14} \rceil$ GB. We then experimentally measured the memory overhead of $\mathcal{S}$ for all database sizes from 2^{10} to 2^{20}. The complete data on calculated and experimental memory overhead can be found in Table 13 of Section C.

For purposes of comparison, holding K *unencrypted* database templates would require $\mathcal{S}$ to store $K \cdot \ell$ floating-point values. Using the same parameters from Table 3, this would incur a memory overhead of $K \cdot 2$ KB for $\mathcal{S}$. Therefore, storing database templates in their encrypted form rather than unencrypted increases the memory overhead of $\mathcal{S}$ by approximately 100x, assuming K exceeds 2^{14}.

Table 4: Comparing Differences between non-FHE-based Secure Face-matching Schemes and HyDia.

Method	Leakage†	TAR loss (%)	Storage/template
IronMask [37]	Probe & Scores	≤4	1MB
Sig.-bit Hash [48]	Auxiliary data	1–6	220B
Talreja et al. [66]	Intermediate binary codes	≤4	**64B**
Zhang et al. [71]	Template ID, Hash digest	<4	560B
SecureFace [46]	Randomized template	≲6	10.5KB
*Biohash/LSH [31, 32, 41]	Template ID, Auxiliary data	2-5	128–2048B
HyDia (ours)	**None**	**<0.1**	192KB

†Information revealed to the server beyond ciphertexts (e.g., probe embeddings, similarity scores, or auxiliary data).
*These methods have been broken via linkage and inversion attacks [24, 53].

While this cost is nonnegligible, it should be highlighted that the memory overhead for storing 2^{20} encrypted templates is approximately 192 GB, which can be handled without issue assuming $\mathcal{S}$ is a large organizational datacenter, as would be the case in many real-world applications of HyDia.

7.3.2 Server RAM Costs. We also performed experiments to record the peak RAM usage incurred by $\mathcal{S}$ in each approach, for all database sizes. We observe that HyDia and *HERS* both peaks, which remain constant across all database sizes, unlike the first three approaches, which could not be scaled to all database sizes due to RAM constraints. We include the full experimental results in Table 14 within Section C. Note that the values in Table 14 represent the maximum RAM usage at any point between both the membership and identification scenarios for a given approach.

7.4 Comparison to non-FHE-based works

Recent non-FHE-based methods for secure face-matching fall into broadly two categories: secure sketch template protection and locality sensitive hashing (LSH), and they exhibit various limitations. Secure-sketch methods such as IronMask [37] and its variants [38] store a hashed codeword plus a user-specific linear transform but reveal every probe and intermediate score to the server. Significant-bit hashing-based methods [48] reduce bandwidth but sacrifice 1–6% TAR (true acceptance rate). Other schemes like [66, 71] support only 1:1 verification and for identification scales linearly in terms of the number of embeddings in the gallery, incurring further accuracy loss from quantisation. In contrast, HyDia does not reveal any data to the server and completes a million-entry encrypted identification in under 100s with an amortized 192 KB for each template. We describe the major qualitative differences of these works compared to HyDia in Table 4.

8 Conclusion

We present HyDia, a novel FHE-based protocol for privacy preserving one-to-many facial matching. Our design addresses key inefficiencies in existing approaches by (1) diagonalizing database vectors to eliminate rotation overhead and introducing a non-rotational inner product algorithm to significantly reduce server-side computation, and (2) employing a hybrid polynomial approximation for secure thresholding to conceal precise similarity scores. We implemented and evaluated HyDia on real-world face embeddings, and HyDia outperforms all state-of-the-art approaches in both computation and communication. Notably, ours remains the only viable option with typical low network bandwidths.

Acknowledgments

This research was supported by the U.S. DHS under Grant 17STQAC 00001-06-00 and NSF under Award No. 2337321. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. DHS or NSF.

We would like to thank Mr. Rasel Ahmed Bhuiyan and Dr. Adam Czajka from the University of Notre Dame for their support in providing real-world datasets for the accuracy experiments. Additionally, we are grateful to Mr. Seunghun Paik from Hanyang University for his insights into non-FHE-based secure face matching works. Finally, we thank the anonymous reviewers for their helpful comments and suggestions.

References

[1] Rashmi Agrawal and Ajay Joshi. 2023. *On Architecting Fully Homomorphic Encryption-based Computing Systems.* Springer Nature Switzerland AG.

[2] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. 2021. Homomorphic encryption standard. *Protecting privacy through homomorphic encryption* (2021), 31–62.

[3] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. 2018. *Homomorphic Encryption Security Standard.* Technical Report. HomomorphicEncryption.org, Toronto, Canada.

[4] Ahmad Al Badawi, Andreea Alexandru, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Carlo Pascoe, Yuriy Polyakov, Ian Quah, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. 2022. OpenFHE: Open-Source Fully Homomorphic Encryption Library. Cryptology ePrint Archive, Paper 2022/915. https://eprint.iacr.org/2022/915 https://eprint.iacr.org/2022/915.

[5] Vishnu Naresh Boddeti. 2018. Secure Face Matching Using Fully Homomorphic Encryption. arXiv:1805.00577 [cs.CV] https://arxiv.org/abs/1805.00577

[6] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. 2018. Threshold cryptosystems from threshold fully homomorphic encryption. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part I 38.* Springer, 565–596.

[7] Rob Bonta. 2022. California consumer privacy act (CCPA). *Retrieved from State of California Department of Justice: https://oag. ca. gov/privacy/ccpa* (2022).

[8] Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. 2020. Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-Sparse Keys. Cryptology ePrint Archive, Paper 2020/1203. https://eprint.iacr.org/2020/1203

[9] Zvika Brakerski. 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings.* Springer, 868–886.

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

[11] Jung Hee Cheon, Hyeongmin Choe, Alain Passelègue, Damien Stehlé, and Elias Suvanto. 2024. Attacks against the IND-CPAD security of exact FHE schemes. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security.* 2505–2519.

[12] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. 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.* Springer, 409–437.

[13] Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. 2019. Efficient Homomorphic Comparison Methods with Optimal Complexity. Cryptology ePrint Archive, Paper 2019/1234. https://eprint.iacr.org/2019/1234

[14] Jung Hee Cheon, Wootae Kim, and Jai Hyun Park. 2022. Efficient Homomorphic Evaluation on Large Intervals. *IEEE Transactions on Information Forensics and Security* 17 (2022), 2553–2568.

[15] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology* 33, 1 (2020), 34–91.

[16] Hyunmin Choi, Jiwon Kim, Chiyoung Song, Simon S Woo, and Hyoungshick Kim. 2024. Blind-Match: Efficient Homomorphic Encryption-Based 1: N Matching for Privacy-Preserving Biometric Identification. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management.* 4423–4430.

[17] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 4690–4699.

[18] Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. 2019. RetinaFace: Single-stage Dense Face Localisation in the Wild. arXiv:1905.00641 [cs.CV] https://arxiv.org/abs/1905.00641

[19] Joshua J. Engelsma, Anil K. Jain, and Vishnu Naresh Boddeti. 2022. HERS: Homomorphically Encrypted Representation Search. arXiv:2003.12197 [cs.CV] https://arxiv.org/abs/2003.12197

[20] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. 2009. Privacy-preserving face recognition. In *Privacy Enhancing Technologies: 9th International Symposium, PETS 2009, Seattle, WA, USA, August 5-7, 2009. Proceedings 9.* Springer, 235–253.

[21] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive* (2012).

[22] Federal Privacy Council. 2025. Fair Information Practice Principles (FIPPs). https://www.fpc.gov/resources/fipps/ Accessed: 2025-02-25.

[23] Craig Gentry. 2009. *A fully homomorphic encryption scheme.* Ph. D. Dissertation. Stanford University. crypto.stanford.edu/craig.

[24] Loubna Ghammam, Koray Karabina, Patrick Lacharme, and Kevin Thiry-Atighehchi. 2020. A cryptanalysis of two cancelable biometric schemes based on index-of-max hashing. *IEEE Transactions on Information Forensics and Security* 15 (2020), 2869–2880.

[25] Vedrana Krivokuća Hahn and Sébastien Marcel. 2022. Biometric template protection for neural-network-based face recognition systems: A survey of methods and evaluation techniques. *IEEE Transactions on Information Forensics and Security* 18 (2022), 639–666.

[26] Shai Halevi and Victor Shoup. 2014. Algorithms in HElib. Cryptology ePrint Archive, Paper 2014/106. https://eprint.iacr.org/2014/106

[27] Shai Halevi and Victor Shoup. 2018. Faster Homomorphic Linear Transformations in HElib. Cryptology ePrint Archive, Paper 2018/244. https://eprint.iacr.org/2018/244

[28] Kyoohyung Han and Dohyeong Ki. 2020. Better bootstrapping for approximate homomorphic encryption. In *Cryptographers' Track at the RSA Conference.* Springer, 364–390.

[29] Alberto Ibarrondo, Hervé Chabanne, Vincent Despiegel, and Melek Önen. 2023. Grote: Group testing for privacy-preserving face identification. In *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy.* 117–128.

[30] InsightFace. 2022. 2D and 3D Face Analysis Project. https://github.com/deepinsight/insightface.

[31] Andrew Teoh Beng Jin, David Ngo Chek Ling, and Alwyn Goh. 2004. Biohashing: two factor authentication featuring fingerprint data and tokenised random number. *Pattern recognition* 37, 11 (2004), 2245–2255.

[32] Zhe Jin, Jung Yeon Hwang, Yen-Lung Lai, Soohyung Kim, and Andrew Beng Jin Teoh. 2017. Ranking-based locality sensitive hashing-enabled cancelable biometrics: Index-of-max hashing. *IEEE Transactions on Information Forensics and Security* 13, 2 (2017), 393–407.

[33] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. In *27th USENIX Security Symposium (USENIX Security 18).* USENIX Association, Baltimore, MD, 1651–1669. https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar

[34] Prabhjot Kaur, Nitin Kumar, and Maheep Singh. 2023. Biometric cryptosystems: a comprehensive survey. *Multimedia Tools and Applications* 82, 11 (2023), 16635–16690.

[35] Tanveer Khan, Alexandros Bakas, and Antonis Michalas. 2021. Blind faith: Privacy-preserving machine learning using function approximation. In *2021 IEEE Symposium on Computers and Communications (ISCC).* IEEE, 1–7.

[36] Mahdi Khosravy, Kazuaki Nakamura, Yuki Hirose, Naoko Nitta, and Noboru Babaguchi. 2022. Model inversion attack by integration of deep generative models: Privacy-sensitive face generation from a face recognition system. *IEEE Transactions on Information Forensics and Security* 17 (2022), 357–372.

[37] Sunpill Kim, Yunseong Jeong, Jinsu Kim, Jungkon Kim, Hyung Tae Lee, and Jae Hong Seo. 2021. IronMask: Modular architecture for protecting deep face template. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 16125–16134.

[38] Sunpill Kim, Hoyong Shin, and Jae Hong Seo. 2025. Deep face template protection in the wild. *Pattern Recognition* 162 (2025), 111336.

[39] Sunpill Kim, Yong Kiam Tan, Bora Jeong, Soumik Mondal, Khin Mi Mi Aung, and Jae Hong Seo. 2024. Scores Tell Everything about Bob: Non-adaptive Face Reconstruction on Face Recognition Systems. In *2024 IEEE Symposium on Security and Privacy (SP).* IEEE, 1684–1702.

[40] Nirajan Koirala, Jonathan Takeshita, Jeremy Stevens, and Taeho Jung. 2024. Summation-based Private Segmented Membership Test from Threshold-Fully Homomorphic Encryption. In *24th Privacy Enhancing Technologies Symposium (PETS 2024)*. Bristol, UK.

[41] Yenlung Lai, Zhe Jin, KokSheik Wong, and Massimo Tistarelli. 2021. Efficient known-sample attack for distance-preserving hashing biometric template protection schemes. *IEEE Transactions on Information Forensics and Security* 16 (2021), 3170–3185.

[42] Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, et al. 2022. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *iEEE Access* 10 (2022), 30039–30054.

[43] Yongwoo Lee, Joon-Woo Lee, Young-Sik Kim, and Jong-Seon No. 2020. Near-optimal polynomial for modulus reduction using l2-norm for approximate homomorphic encryption. *IEEE Access* 8 (2020), 144321–144330.

[44] Baiyu Li and Daniele Micciancio. 2021. On the security of homomorphic encryption on approximate numbers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 648–677.

[45] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2018. SphereFace: Deep Hypersphere Embedding for Face Recognition. arXiv:1704.08063 [cs.CV] https://arxiv.org/abs/1704.08063

[46] Guangcan Mai, Kai Cao, Xiangyuan Lan, and Pong C Yuen. 2020. Secureface: Face template protection. *IEEE Transactions on Information Forensics and security* 16 (2020), 262–277.

[47] Nicole Martinez-Martin. 2019. What Are Important Ethical Implications of Using Facial Recognition Technology in Health Care? *AMA Journal of Ethics* 21 (2019), 180–187.

[48] Deen Dayal Mohan, Nishant Sankaran, Sergey Tulyakov, Srirangaraj Setlur, and Venu Govindaraju. 2019. Significant feature based representation for template protection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2389–2396.

[49] Department of Homeland Security. 2017. Handbook for Safeguarding Sensitive PII. https://www.dhs.gov/sites/default/files/2024-03/17_1204_priv_handbooksafeguardingsensitivepii_rev3_047-01-007.pdf

[50] Department of Homeland Security. 2021. Office of Biometric Identity Management Identification Services. https://www.dhs.gov/obim-biometric-identification-services

[51] Department of Homeland Security. 2022. DHS Use Cases of Privacy Enhancing Technologies. https://pets4hse.org/PETS4HSEUseCases.pdf

[52] OpenFHE . 2024. UnitTestCKKSrns.cpp – Re-Encryption Test in OpenFHE. https://github.com/openfheorg/openfhe-development/blob/main/src/pke/unittest/utckksrns/UnitTestCKKSrns.cpp. Accessed: 2025-05-24.

[53] Seunghun Paik, Sunpill Kim, and Jae Hong Seo. 2023. Security Analysis on Locality-Sensitive Hashing-based Biometric Template Protection Schemes.. In *BMVC*. 535–537.

[54] Vishal M Patel, Nalini K Ratha, and Rama Chellappa. 2015. Cancelable biometrics: A review. *IEEE signal processing magazine* 32, 5 (2015), 54–65.

[55] P Jonathon Phillips, Patrick J Flynn, Todd Scruggs, Kevin W Bowyer, Jin Chang, Kevin Hoffman, Joe Marques, Jaesik Min, and William Worek. 2005. Overview of the face recognition grand challenge. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1. IEEE, 947–954.

[56] Yuriy Polyakov. 2024. Which Operations can be followed Multiplication w/o Relinearization? https://openfhe.discourse.group/t/which-operations-can-be-followed-multiplication-w-o-relinearization/1148

[57] Yuriy Polyakov, Kurt Rohloff, Gyana Sahu, and Vinod Vaikuntanathan. 2017. Fast proxy re-encryption for publish/subscribe systems. *ACM Transactions on Privacy and Security (TOPS)* 20, 4 (2017), 1–31.

[58] Lorenzo Rovida and Alberto Leporati. 2024. Encrypted image classification with low memory footprint using fully homomorphic encryption. *Cryptology ePrint Archive* (2024).

[59] Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. 2009. Efficient privacy-preserving face recognition. In *International conference on information security and cryptology*. Springer, 229–244.

[60] Alamgir Sardar, Saiyed Umer, Ranjeet Kumar Rout, and Muhammad Khurram Khan. 2022. A secure and efficient biometric template protection scheme for palmprint recognition system. *IEEE Transactions on Artificial Intelligence* 4, 5 (2022), 1051–1063.

[61] Nigel P Smart and Frederik Vercauteren. 2014. Fully homomorphic SIMD operations. *Designs, codes and cryptography* 71 (2014), 57–81.

[62] Zhigang Song, Gong Wang, Wenqin Yang, Yunliang Li, Yinsheng Yu, Zeli Wang, Xianghan Zheng, and Yang Yang. 2025. Privacy-preserving method for face recognition based on homomorphic encryption. *PloS one* 20, 2 (2025), e0314656.

[63] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2014. Deep Learning Face Representation by Joint Identification-Verification. arXiv:1406.4773 [cs.CV] https://arxiv.org/abs/1406.4773

[64] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 1701–1708. https://doi.org/10.1109/CVPR.2014.220

[65] Jonathan Takeshita, Colin McKechney, Justin Pajak, Antonis Papadimitriou, Ryan Karl, and Taeho Jung. 2021. Gps: Integration of graphene, palisade, and sgx for large-scale aggregations of distributed data. *Cryptology ePrint Archive* (2021).

[66] Veeru Talreja, Matthew C Valenti, and Nasser M Nasrabadi. 2019. Zero-shot deep hashing and neural network based error correction for face template protection. In *2019 IEEE 10th International Conference on Biometrics Theory, Applications and Systems (BTAS)*. IEEE, 1–10.

[67] U.S. Department of Homeland Security. 2025. 2024 Update on DHS's Use of Face Recognition and Face Capture Technologies. https://www.dhs.gov/archive/news/2025/01/16/2024-update-dhss-use-face-recognition-face-capture-technologies. Accessed: 2025-02-24.

[68] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A practical guide, 1st ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.

[69] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. 2018. CosFace: Large Margin Cosine Loss for Deep Face Recognition. arXiv:1801.09414 [cs.CV] https://arxiv.org/abs/1801.09414

[70] Philippe Weinzaepfel, Hervé Jégou, and Patrick Pérez. 2011. Reconstructing an image from its local descriptors. In *CVPR 2011*. IEEE, 337–344.

[71] Kaiyi Zhang, Hongrui Cui, and Yu Yu. 2021. Facial template protection via lattice-based fuzzy extractors. *Cryptology ePrint Archive* (2021).

[72] Zheng Zhu, Guan Huang, Jiankang Deng, Yun Ye, Junjie Huang, Xinze Chen, Jiagang Zhu, Tian Yang, Jiwen Lu, Dalong Du, et al. 2021. Webface260m: A benchmark unveiling the power of million-scale deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10492–10502.

A Correctness Proofs

Lemma 1 (Diagonal Matrix-Vector Multiplication For Square Matrices, Halevi and Shoup [26]). *Let $A \in \mathbb{R}^{n\times n}$ be a square matrix represented in diagonal order by vectors $d_0, \ldots, d_{n-1}$, where $d_i[j] = A[j][j+i \bmod n]$ for all $i, j \in \{0, 1, \ldots, n-1\}$. Then for any vector $v \in \mathbb{R}^n$, the matrix-vector product $w = Av$ can be computed as:*

$$w = \sum_{i=0}^{n-1} d_i \odot (v \lll i) \tag{1}$$

where $\odot$ denotes element-wise multiplication and $(v \lll i)$ denotes v rotated left by i positions.

To show $w = Av$, it is equivalent to prove $w \leftarrow \sum_{i=0}^{n-1} d_i \odot (v \lll i)$. We can prove it as follows:

$$\begin{aligned} w[j] &= \sum_{i=0}^{n-1} d_i[j] \cdot (v \lll i)[j] \\ &= \sum_{i=0}^{n-1} A[j][j+i \bmod n] \cdot v[j+i \bmod n] \\ &= \sum_{k=0}^{n-1} A[j][k] \cdot v[k] \\ &= Av. \end{aligned}$$

A more detailed proof of the diagonal matrix-vector multiplication (DMVM) for square matrices (Lemma 1) can be found in Section 4.3 of [26]. We extend Lemma 1 for the rectangular variant used in HyDia in *Lemma* 2 below.

Lemma 2 (Diagonal Matrix-Vector Multiplication For Rectangular Matrices). *Let $A \in \mathbb{R}^{n\times l}$ be a rectangular matrix where l divides n, and D be a diagonalization of A via Algorithm 2. Then for any $v \in \mathbb{R}^l$, the output of Algorithm 3 on inputs D and v is equal to Av.*

Proof. First, we show for all $j = 0, \ldots, n-1$,

$$\left(\sum_{i=0}^{\ell-1} D[i] \odot (v' \lll i)\right)[j] = (Av)[j]. \tag{2}$$

This can be seen as follows:

$$\left(\sum_{i=0}^{\ell-1} D[i] \odot (v' \lll i)\right)[j] \tag{3}$$

$$= \sum_{i=0}^{\ell-1} D[i][j] \cdot (v' \lll i)[j] \tag{4}$$

$$= \sum_{i=0}^{\ell-1} A[j][i+j \bmod \ell] \cdot (v' \lll i)[j] \tag{5}$$

$$= \sum_{i=0}^{\ell-1} A[j][i+j \bmod \ell] \cdot v'[i+j \bmod n] \tag{6}$$

$$= \sum_{i=0}^{\ell-1} A[j][i+j \bmod \ell] \cdot v[(i+j \bmod n) \bmod \ell] \tag{7}$$

$$= \sum_{i=0}^{\ell-1} A[j][i+j \bmod \ell] \cdot v[i+j \bmod \ell] \tag{8}$$

$$= \sum_{m=0}^{\ell-1} A[j][m] \cdot v[m] \tag{9}$$

$$= (Av)[j]. \tag{10}$$

Therefore, $w \leftarrow Av$, proving the correctness of the expression. □

Proof of Theorem 1. To prove the correctness of HyDia, we analyze each step and show that the final output correctly identifies matches based on the cosine similarity threshold θ.

For any non-zero vector $\vec{v}$, the normalized vector $\bar{v} = \vec{v}/||\vec{v}||$ has unit length. Consequently, for any two normalized vectors $\bar{d}_i$ (database vector) and $\bar{q}$ (query vector), their dot product $\bar{d}_i \cdot \bar{q}$ equals the similarity score $cos(\vec{d}_i, \vec{q})$.

In Lemma 2, we have shown that $\mathbf{q}$ and a diagonalization D of database vectors A (computed via Algorithm 2) will subsequently produce the correct similarity scores $\mathbf{S_0} = A\mathbf{q}$ via Algorithm 3.

Algorithm 4 correctly identifies similarity scores $\mathbf{S_0}$ exceeding a given threshold θ within the allotted multiplicative depth κ as demonstrated in Figure 4. This ensures the comparison results correctly reflect whether $\cos(\vec{d}_i, \vec{q}) \geq \theta$ for each database vector.

Finally, the aggregation operations preserve these comparison results while correctly formatting them for each scenario. For the Membership scenario, the server homomorphically adds all comparison ciphertexts to the first ciphertext. Since our hybrid approximation function outputs positive values only for similarities exceeding the threshold θ, the sum $v = 0$ if and only if no matches occur.

For the Identification scenario, by returning the entire vector of comparison ciphertexts without summation, the protocol preserves the individual match status of each database entry. Thus, after decryption, the client can directly observe which specific indices have values $v_i \geq 1$, thus correctly identifying all database vectors in which the similarity exceeds the given threshold.

Therefore, in both scenarios, the decrypted values correctly reflect the matching criteria defined by the cosine similarity threshold θ. □

B Security Proofs

We prove Theorem 2 by constructing a series of hybrid distributions and showing their indistinguishability through a sequence of lemmas. Then, we gradually replace real protocol components with simulated ones, demonstrating that each transition preserves the computational indistinguishability.

Consider the following hybrid distributions:

$H_0 = (\mathbf{M}, \mathbf{q}, \mathbf{ct})$ // Real world

$H_1 = (\mathsf{SIM}.\mathbf{M}, \mathbf{q}, \mathbf{ct})$ // First hybrid

$H_2 = (\mathsf{SIM}.\mathbf{M}, \mathsf{SIM}.\mathbf{q}, \mathbf{ct})$ // Second hybrid

$H_3 = (\mathsf{SIM}.\mathbf{M}, \mathsf{SIM}.\mathbf{q}, \mathsf{SIM}.\mathbf{ct})$ // Final simulation

To simulate the view of a semi-honest server, we define the simulator in Algorithm 6.

It is important to note that SIM runs in polynomial time in λ. This follows from the fact that sampling R_D, r_q, r_s requires $O(ld)$ random bits, and the Encrypt operation runs in $\text{poly}(\lambda)$ time. Consequently, the total runtime of the simulator is bounded by $O(ld \cdot \text{poly}(\lambda))$, which is polynomial in the security parameter.

Lemma 3 (Indistinguishability of H_0 and H_1). *If there exists a PPT distinguisher $\mathcal{D}$ that can distinguish between H_0 and H_1 with non-negligible advantage ϵ, then there exists an adversary $\mathcal{B}_1$ that breaks the IND-CPA security of CKKS with advantage ϵ.*

Proof. We construct adversary $\mathcal{B}_1$ as follows. First, $\mathcal{B}_1$ initiates a CKKS security challenge by choosing messages $m_0 = D$ and $m_1 = R_D$, receiving a challenge ciphertext c^* from the CKKS challenger. Then, $\mathcal{B}_1$ constructs a view for $\mathcal{D}$ by setting $\mathbf{M} = c^*$ and honestly generating $\mathbf{q}$ and $\mathbf{ct}$. Finally, $\mathcal{B}_1$ runs $\mathcal{D}$ on this view and outputs whatever $\mathcal{D}$ outputs.

For the analysis, observe that if c^* encrypts m_0, $\mathcal{D}$ sees distribution H_0, and if c^* encrypts m_1, $\mathcal{D}$ sees distribution H_1. Therefore, $\mathcal{B}_1$'s advantage in breaking CKKS is exactly $\mathcal{D}$'s advantage. □

Lemma 4 (Indistinguishability of H_1 and H_2). *If there exists a PPT distinguisher $\mathcal{D}$ that can distinguish between H_1 and H_2 with non-negligible advantage ϵ, then there exists an adversary $\mathcal{B}_2$ that breaks the IND-CPA security of CKKS with advantage ϵ.*

Proof. We construct adversary $\mathcal{B}_2$ as follows. $\mathcal{B}_2$ initiates a CKKS security challenge by choosing messages $m_0 = q$ and $m_1 =$

Algorithm 6 $\mathsf{SIM}(1^\lambda, \text{aux})$

Input: Security parameter λ, auxiliary information aux
Output: Simulated view $\text{VIEW}_{\mathsf{SIM}}$
Sample $R_D \xleftarrow{\$} \mathbb{Z}_q^{l \times d}$ ▷ l database size, d vector dimension
Sample $r_q \xleftarrow{\$} \mathbb{Z}_q^d$
Sample $r_s \xleftarrow{\$} \mathbb{Z}_q^l$
$\mathsf{SIM}.\mathbf{M} \leftarrow \mathsf{Encrypt}(pk, R_D)$
$\mathsf{SIM}.\mathbf{q} \leftarrow \mathsf{Encrypt}(pk, r_q)$
$\mathsf{SIM}.\mathbf{ct} \leftarrow \mathsf{Encrypt}(pk, r_s)$
return $\text{VIEW}_{\mathsf{SIM}} = (\mathsf{SIM}.\mathbf{M}, \mathsf{SIM}.\mathbf{q}, \mathsf{SIM}.\mathbf{ct})$

r_q, receiving a challenge ciphertext c^* from the CKKS challenger. Then, $\mathcal{B}_2$ constructs a view for $\mathcal{D}$ by first generating a random R_D and setting SIM.$\mathbf{M}$ = Encrypt(R_D), setting $\mathbf{q} = c^*$, and honestly generating $\mathbf{ct}$. Finally, $\mathcal{B}_2$ runs $\mathcal{D}$ on this view and outputs whatever $\mathcal{D}$ outputs.

For the analysis, observe that if c^* encrypts m_0, $\mathcal{D}$ sees distribution H_1, and if c^* encrypts m_1, $\mathcal{D}$ sees distribution H_2. Therefore, $\mathcal{B}_2$'s advantage in breaking CKKS is exactly $\mathcal{D}$'s advantage. □

Lemma 5 (Indistinguishability of H_2 and H_3). *If there exists a PPT distinguisher $\mathcal{D}$ that can distinguish between H_2 and H_3 with non-negligible advantage ϵ, then there exists an adversary $\mathcal{B}_3$ that breaks the IND-CPA security of CKKS with advantage ϵ.*

Proof. We construct adversary $\mathcal{B}_3$ as follows. $\mathcal{B}_3$ initiates a CKKS security challenge by choosing messages $m_0 = \cos(D, q)$ and $m_1 = r_s$, receiving a challenge ciphertext c^* from the CKKS challenger. Then, $\mathcal{B}_3$ constructs a view for $\mathcal{D}$ by first generating a random R_D and setting SIM.$\mathbf{M}$ = Encrypt(R_D), generating a random r_q and setting SIM.$\mathbf{q}$ = Encrypt(r_q), and setting $\mathbf{ct} = c^*$. Finally, $\mathcal{B}_3$ runs $\mathcal{D}$ on this view and outputs whatever $\mathcal{D}$ outputs.

For the analysis, observe that if c^* encrypts m_0, $\mathcal{D}$ sees distribution H_2, and if c^* encrypts m_1, $\mathcal{D}$ sees distribution H_3. Therefore, $\mathcal{B}_3$'s advantage in breaking CKKS is exactly $\mathcal{D}$'s advantage. □

Proof of Theorem 1. We have shown through Lemmas 1-3 that each consecutive pair of hybrids is computationally indistinguishable under the IND-CPA security of CKKS. By the triangle inequality:

$$\begin{aligned}|\Pr[\mathcal{D}(H_0)=1] - \Pr[\mathcal{D}(H_3)=1]| &\le |\Pr[\mathcal{D}(H_0)=1] - \Pr[\mathcal{D}(H_1)=1]| + \\ &\quad |\Pr[\mathcal{D}(H_1)=1] - \Pr[\mathcal{D}(H_2)=1]| + \\ &\quad |\Pr[\mathcal{D}(H_2)=1] - \Pr[\mathcal{D}(H_3)=1]| \\ &\le \mathrm{negl}_1(\lambda) + \mathrm{negl}_2(\lambda) + \mathrm{negl}_3(\lambda) \\ &= \mathrm{negl}(\lambda)\end{aligned}$$

Since H_0 represents the real protocol execution and H_3 is exactly the output of our simulator SIM, we have shown that:

$$|\Pr[\mathcal{D}(\mathsf{VIEW}_{\mathsf{real}})=1] - \Pr[\mathcal{D}(\mathsf{VIEW}_{\mathsf{SIM}})=1]| \le \mathrm{negl}(\lambda)$$

Therefore, the real protocol execution is computationally indistinguishable from the simulated view, completing our security proof. □

C Complete Experimental Data

Figure 7 shows the similarity score distribution of all pairs of query and database vectors generated by partitioning our dataset. As described in Section 7, we partitioned our dataset randomly into 50 query vectors and 44,228 database vectors, resulting in 2,211,400 unique query-database pairs. The score distribution of pairs, which were labeled as matching identities, are shown in green, and their frequencies correspond to the right axis ticks. The score distribution of pairs, which were labeled as distinct identities, are shown in red, and their frequencies correspond to the left axis ticks.

Table 5 includes the experimental client overhead values for each approach. Note that $N_{in} = 4$ was chosen to be the optimal subvector parameter by Choi *et al.* [16].

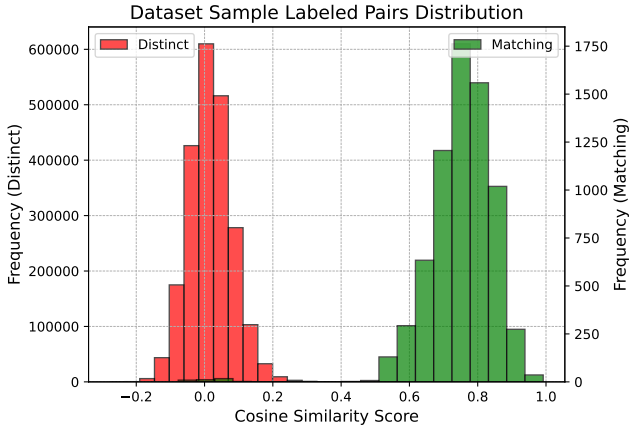

Figure 7: Cosine Similarity Distribution of Labeled Pairs in Random Dataset Partition

Table 5: Client Overhead for All Approaches (seconds)

Approach	Client Overhead
Baseline	0.083
GROTE	0.099
Blind-Match ($N_{in} = 4$)	0.165
HERS ($\ell = 512$)	2.540
HyDia	**0.033**

Table 6: Server Overhead for Membership and Identification Scenarios across Database Sizes (seconds)

DB	Baseline		*GROTE*		*Blind-Match*		*HERS*		HyDia	
	Memb.	Id.	Memb.	Id.	Memb.	Id.	Memb.	Id.	Memb.	Id.
2^{10}	11.5	11.3	17.5	20.6	8.5	8.1	**4.3**	**3.5**	5.9	4.7
2^{11}	17.1	16.8	25.2	28.3	13.0	12.8	**4.2**	**3.5**	4.5	4.2
2^{12}	35.9	35.6	48.7	50.7	22.0	21.0	**4.1**	**3.5**	4.5	4.2
2^{13}	64.6	64.2	89.6	92.1	40.1	39.5	**4.2**	**3.3**	4.5	4.4
2^{14}	133.8	135.8	175.2	176.0	79.7	80.0	**4.2**	**3.5**	4.5	4.2
2^{15}	264.7	265.5	338.4	336.7	161.1	161.2	**6.2**	**5.4**	**6.2**	5.9
2^{16}	–	–	–	–	–	–	9.8	9.1	**9.6**	**9.0**
2^{17}	–	–	–	–	–	–	17.0	15.8	**16.0**	**15.5**
2^{18}	–	–	–	–	–	–	31.0	28.8	**28.7**	**27.6**
2^{19}	–	–	–	–	–	–	58.2	54.0	**53.2**	**49.7**
2^{20}	–	–	–	–	–	–	115.5	110.9	**102.4**	**96.5**

Table 7: End-to-End Overhead for All Approaches over 2^{15} Database Subjects (seconds)

Network	Baseline		*GROTE*		*Blind-Match*		*HERS*		HyDia	
Bandwidth	Memb.	Id.	Memb.	Id.	Memb.	Id.	Memb.	Id.	Memb.	Id.
64Kbps	3849.613	3850.5	5203.517	7634.3	8483.286	8483.3	394181.932	394949.5	**1542.931**	**2311.0**
2Mbps	376.8	377.6	490.5	564.9	421.4	421.4	12326.7	12349.8	**54.2**	**77.9**
1Gbps	265.0	265.8	338.8	337.3	161.8	161.9	32.8	32.0	**6.3**	**6.0**
20Gbps	264.8	265.6	338.5	336.9	161.3	161.4	10.0	9.1	**6.2**	**5.9**

Table 6 presents the server overhead computational times for both the Membership and Indentification scenarios, which are plotted in Figure 6(a) and Figure 6(b). We computed this data by averaging 10 trial runs for each approach on each database size. As described in Section 7, three of the approaches could not scale beyond a database size of 2^{15}, and therefore we omit the results for these approaches.

Table 7 displays the end-to-end query times of different approaches over different network bandwidth values, which are plotted in Figure 6(c) and Figure 6(d) in Section 7. Table 8 additionally shows the end-to-end query times of *HERS* and HyDia when

Table 8: End-to-End Overhead of State-of-the-Art Approaches over 2^{20} Database Subjects (seconds)

Network	*HERS*		HyDia	
Bandwidth	Membership	Identification	Membership	Identification
64Kbps	394291.306	442694.141	**1639.188**	**50040.730**
2Mbps	12436.037	13944.365	**150.474**	**1657.508**
1Gbps	142.183	140.740	**102.544**	**99.808**
20Gbps	119.328	115.078	**102.455**	**96.912**

Table 9: Transmission Size of All Approaches over 2^{15} Database Subjects (MBs)

Scenario	Baseline	*GROTE*	*Blind-Match*	*HERS*	HyDia
Membership	28.007	38.008	65.015	3079.478	**12.006**
Identification	28.007	57.012	65.015	3085.481	**18.009**

Table 10: Communication Overhead of All Approaches over 2^{15} Database Subjects (seconds)

Network	Baseline		*GROTE*		*Blind-Match*		*HERS*		HyDia	
Bandwidth	Memb.	Id.	Memb.	Id.	Memb.	Id.	Memb.	Id.	Memb.	Id.
64Kbs	3584.8	3584.8	4865.0	7297.5	8322.0	8322.0	394173.2	394941.5	**1536.7**	**2305.1**
2Mbps	112.0	112.0	152.0	228.0	260.1	260.1	12317.9	12341.9	**48.0**	**72.0**
1Gbps	0.2	0.2	0.3	0.4	0.5	0.5	24.1	24.1	**0.1**	**0.1**
20Gbps	0.01	0.01	0.01	0.02	0.03	0.03	1.20	1.21	**0.00**	**0.01**

Table 11: Server Overhead Comparison Between HyDia without Hoisting Optimization and With (seconds)

DB	Non-Hoisted Memb.	Hoisted Memb.	Memb. Diff.	Non-Hoisted Id.	Hoisted Id.	Id. Diff.
2^{10}	11.1	**5.9**	5.2	9.7	**4.7**	5.0
2^{11}	10.7	**4.5**	6.2	9.9	**4.2**	5.7
2^{12}	10.8	**4.5**	6.2	9.9	**4.2**	5.7
2^{13}	10.6	**4.5**	6.1	9.9	**4.4**	5.6
2^{14}	10.7	**4.5**	6.2	10.0	**4.2**	5.8
2^{15}	12.1	**6.2**	5.9	11.4	**5.9**	5.5
2^{16}	14.9	**9.6**	5.4	14.2	**9.0**	5.2
2^{17}	21.0	**16.0**	5.0	20.3	**15.5**	4.8
2^{18}	32.8	**28.7**	4.1	32.1	**27.6**	4.5
2^{19}	56.7	**53.2**	3.5	55.8	**49.7**	6.1
2^{20}	119.1	**102.4**	16.7	109.8	**96.5**	13.3

there are 2^{20} database subjects. As described in Section 7, other approaches could not be scaled to these many subjects and are thus omitted in the comparison. Finally, Table 9 shows the ciphertext communication size in the membership and identification scenario for all methods, and Table 10 shows the communication latency for transmitting these query ciphertexts over various network bandwidths.

D Hoisting Optimization for Homomorphic Rotations

The "hoisting" optimization, introduced by Halevi and Shoup [27], is applied in Section 5 to reduce the computational overhead of rotating the same ciphertext by multiple different factors. At a high level, Halevi and Shoup identified that multiple rotations upon the same ciphertext would each begin with a common sub-operation, independent of the specific rotation factor. Therefore, that sub-operation could be precomputed exactly once, and each separate rotation could use that precomputed result without needing to individually compute it. In practice, this significantly reduces the number of operations required across all rotations [33]. The optimization is referred to as "hoisting" since, if many rotations upon the same ciphertext were to be performed in a loop, this optimization would hoist all common sub-operations out of that loop [27].

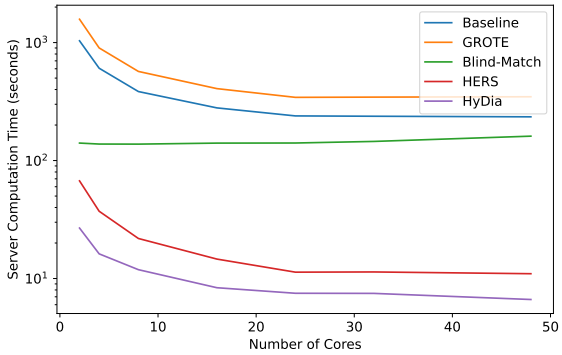

Figure 8: Membership Scenario Server Overhead over 2^{15} Database Subjects using Varying Cores

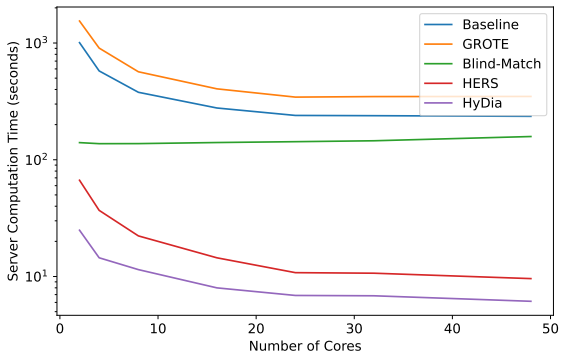

Figure 9: Identification Scenario Server Overhead over 2^{15} Database Subjects using Varying Cores

Table 12: Server Overhead of each Approach over 2^{15} Database Subjects across Different Numbers of Cores (seconds)

	Baseline		*GROTE*		*Blind-Match*		*HERS*		HyDia	
Cores	Memb.	Id.	Memb.	Id.	Memb.	Id.	Memb.	Id.	Memb.	Id.
2	1035.4	1007.2	1577.5	1543.2	140.7	140.2	67.3	66.8	**26.8**	**24.9**
4	606.3	576.7	900.7	904.7	138.0	137.3	37.2	36.9	**16.2**	**14.5**
8	384.5	378.4	569.4	567.1	137.8	137.5	21.8	22.3	**11.9**	**11.5**
16	279.5	277.6	407.1	405.1	140.5	140.5	14.6	14.5	**8.4**	**8.0**
24	239.0	239.7	343.3	344.1	140.8	142.9	11.3	10.8	**7.5**	**6.9**
32	237.4	238.5	344.9	347.7	145.2	145.4	11.4	10.7	**7.5**	**6.8**
48	234.8	235.5	347.3	348.6	161.0	158.2	11.0	9.6	**6.6**	**6.1**

From a technical perspective, the hoisting optimization works because the rotation operation in CKKS is an automorphism ϕ_k, where k is the rotation factor. By definition, this automorphism distributes over addition and multiplication, and also commutes with the RNS decomposition of a CKKS ciphertext [8].

When this automorphism is applied to a ciphertext $\mathbf{ct}' \leftarrow \phi_k(\mathbf{ct})$, it can only be properly decrypted by $\phi_k(sk)$, and key-switching must be performed so that $\mathbf{ct}'$ can be decrypted by the original private key sk. Therefore, a typical rotation operation in CKKS involves three main sub-operations [27]:

(1) The automorphism is applied to the ciphertext.

Table 13: Server Memory Costs for Storing Databases in Encrypted HyDia Form versus Unencrypted Form (GB)

DB	HyDia (Experimental)	HyDia (Calculated)	Unencrypted (Calculated)
2^{10}	3.001	3.000	0.002
2^{11}	3.001	3.000	0.004
2^{12}	3.001	3.000	0.008
2^{13}	3.001	3.000	0.016
2^{14}	3.001	3.000	0.031
2^{15}	6.003	6.000	0.063
2^{16}	12.006	12.000	0.125
2^{17}	24.012	24.000	0.250
2^{18}	48.023	48.000	0.500
2^{19}	96.047	96.000	1.000
2^{20}	192.093	192.000	2.000

Table 14: Server Peak RAM Costs for Membership and Index Scenarios (GB)

DB	Baseline	*GROTE*	*Blind-Match*	*HERS*	HyDia
2^{10}	58.2	80.7	53.3	**31.8**	33.3
2^{11}	60.2	83.3	53.3	**31.8**	33.3
2^{12}	62.3	86.4	53.5	**31.8**	33.3
2^{13}	62.7	87.1	53.9	**31.8**	33.3
2^{14}	64.5	89.2	54.4	**31.8**	33.3
2^{15}	67.2	93.3	54.8	**31.9**	33.3
2^{16}	–	–	–	**31.9**	33.3
2^{17}	–	–	–	**31.9**	33.3
2^{18}	–	–	–	**31.9**	33.3
2^{19}	–	–	–	**31.9**	33.3
2^{20}	–	–	–	**31.9**	33.3

(2) RNS decomposition is performed upon the ciphertext. This is the most expensive sub-operation [26].
(3) Key-switching is performed upon the decomposed ciphertext.

Halevi and Shoup noted that the RNS decomposition of a ciphertext is fully independent of the factor it is being rotated by. Furthermore, since ϕ_k is an automorphism, it can be applied directly upon the RNS decomposition of a ciphertext, and the rotation procedure would remain correct. Therefore, the hoisting optimization involves switching the order of steps (1) and (2) from above [27]. Now that the RNS decomposition is the first step in the rotation procedure, that decomposition can be precomputed for a ciphertext exactly once and reused by all subsequent rotations of that ciphertext, even if those rotations are by different factors.

It should be noted that the hoisting optimization is only applicable when the *same* ciphertext needs to be rotated by multiple different rotation factors. This is because the RNS decomposition for one ciphertext cannot be used in the rotation procedure of another ciphertext. As such, this optimization is only applicable to HyDia at the beginning of the score computation step performed by $\mathcal{S}$, where the same query ciphertext must be rotated $\ell - 1$ times.

E Complexity Analysis

We present the algorithmic complexities required by $\mathcal{S}$ to compute cosine similarity scores in the baseline approach, and in HyDia. These complexities are given with respect to the number of database template vectors K, and involve constant system parameters N and ℓ. We will demonstrate that, due to its diagonal encoding of database template vectors, HyDia is able to compute cosine similarity scores with far fewer expensive homomorphic operations, particularly relinearizations, rescalings, and rotations. Furthermore, regarding all other homomorphic operations, the complexity of HyDia is either less than or equal to the baseline approach.

It should be noted that we only discuss the complexities of the score-computation step, which is equivalent to matrix-vector multiplication, in this section. The score-comparison step, as well as the operations of C, have equivalent complexities between the baseline approach and HyDia, and therefore are not included here.

E.1 Complexity of Baseline Approach

Recall that in the baseline approach, database template vectors are not encoded diagonally, but instead packed sequentially into ciphertexts. As such, the baseline approach stores K database template vectors within $\lceil 2K\ell/N \rceil$ ciphertexts. To compute cosine similarity scores, $\mathcal{S}$ must perform Algorithm 1 upon each database ciphertext. In sum, $\mathcal{S}$ must perform $\lceil 2K\ell/N \rceil$ ciphertext-ciphertext multiplications, and $\lceil 2K\ell/N \rceil \cdot \log_2(\ell)$ of both ciphertext-ciphertext additions and rotations.

Additionally, recall that Algorithm 1 returns sparsely-packed ciphertexts, with scores positioned at intervals of ℓ. Therefore, the baseline approach requires a score-merge operation to remove non-score values and rearrange valid scores into a dense packing. This operation requires a ciphertext-plaintext multiplication for each of the $\lceil 2K\ell/N \rceil$ ciphertexts, to mask non-score values. After masking, additions and rotations are used to combine ℓ sparsely-packed ciphertexts into one densely-packed ciphertext. Therefore, this score-merge operation also requires $\lceil 2K\ell/N \rceil - \lceil 2K/N \rceil$ rotations and ciphertext-ciphertext additions. After all scores have been densely packed, those resulting ciphertexts must be relinearized and rescaled, involving $\lceil 2K/N \rceil$ operations each.

E.2 Complexity of HyDia

In HyDia, before $\mathcal{S}$ can begin computing cosine similarity scores using Algorithm 3, it must compute and store all ℓ unique rotations of the query ciphertext, as defined by Algorithm 5. Since it is given the unrotated query ciphertext by C, it must perform $\ell - 1$ rotations to obtain the rest.

In order for $\mathcal{S}$ to compute K similarity scores following HyDia, it must perform Algorithm 3 upon each grouping of ℓ ciphertexts. Since our diagonal encoding can pack up to $N/2$ templates into each grouping of ℓ ciphertexts, $\mathcal{S}$ must perform Algorithm 3 $\lceil 2K/N \rceil$ times. Each execution of Algorithm 3 requires ℓ ciphertext-ciphertext multiplications, followed by $\ell - 1$ ciphertext-ciphertext additions, one relinearization, and one rescaling.

Recall that Algorithm 3 outputs a vector of cosine similarity scores, which are fully packed in sequential order. Therefore, unlike the baseline approach, HyDia does not require a score-merge operation to rearrange the scores into packed, sequential order.

Table 15: Algorithmic Complexity for Cosine Similarity Score Computation

Approach	Cipher-Cipher Addition	Cipher-Cipher Mult.	Cipher-Plain Mult.	Relin.	Rescale	Standard Rotation	Hoisted Rotation
Baseline (Inner Product)	$\lceil 2K\ell/N \rceil \cdot \log_2(\ell)$	$\lceil 2K\ell/N \rceil$	0	$\lceil 2K\ell/N \rceil$	$\lceil 2K\ell/N \rceil$	$\lceil 2K\ell/N \rceil \cdot \log_2(\ell)$	0
Baseline (Score-Merge)	$\lceil 2K\ell/N \rceil - \lceil 2K/N \rceil$	0	$\lceil 2K\ell/N \rceil$	$\lceil 2K/N \rceil$	$\lceil 2K/N \rceil$	$\lceil 2K\ell/N \rceil - \lceil 2K/N \rceil$	0
HyDia	$\lceil 2K/N \rceil \cdot (\ell - 1)$	$\lceil 2K/N \rceil \cdot \ell$	0	$\lceil 2K/N \rceil$	$\lceil 2K/N \rceil$	0	$\ell - 1$