Blind-Match: Efficient Homomorphic Encryption-Based 1:N Matching for Privacy-Preserving Biometric Identification

Hyunmin Choi
NAVER Cloud
Seongnam, Republic of Korea
hyunmin.choi@navercorp.com

Jiwon Kim
Sungkyunkwan University
Suwon, Republic of Korea
merwl0@g.skku.edu

Chiyoung Song
NAVER Cloud
Seongnam, Republic of Korea
chiyoung.song@navercorp.com

Simon S. Woo*
Sungkyunkwan University
Suwon, Republic of Korea
swoo@g.skku.edu

Hyoungshick Kim*
Sungkyunkwan University
Suwon, Republic of Korea
hyoung@skku.edu

Abstract

We present *Blind-Match*, a novel biometric identification system that leverages homomorphic encryption (HE) for efficient and privacy-preserving 1:N matching. *Blind-Match* introduces a HE-optimized cosine similarity computation method, where the key idea is to divide the feature vector into smaller parts for processing rather than computing the entire vector at once. By optimizing the number of these parts, *Blind-Match* minimizes execution time while ensuring data privacy through HE. *Blind-Match* achieves superior performance compared to state-of-the-art methods across various biometric datasets. On the LFW face dataset, *Blind-Match* attains a 99.63% Rank-1 accuracy with a 128-dimensional feature vector, demonstrating its robustness in face recognition tasks. For fingerprint identification, *Blind-Match* achieves a remarkable 99.55% Rank-1 accuracy on the PolyU dataset, even with a compact 16-dimensional feature vector, significantly outperforming the state-of-the-art method, Blind-Touch, which achieves only 59.17%. Furthermore, *Blind-Match* showcases practical efficiency in large-scale biometric identification scenarios, such as Naver Cloud's FaceSign, by processing 6,144 biometric samples in 0.74 seconds using a 128-dimensional feature vector.

CCS Concepts

• **Security and privacy → Software and application security**.

Keywords

Biometric Identification, Homomorphic Encryption, Privacy

ACM Reference Format:
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 (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3627673.3680017

*Corresponding authors.

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3680017

1 Introduction

Biometric identification, such as fingerprint, facial, and iris recognition, is commonly used for user authentication on personal devices [10]. However, its adoption in web and cloud environments is limited due to the difficulty in changing or revoking biometric data once compromised, as demonstrated by the 2016 US Office of Personnel Management breach, where 5.6 million individuals' fingerprints were stolen [18]. This incident highlights the critical need for secure management of biometric data on servers.

Biometric template protection (BTP) techniques, such as locality-sensitive hashing (LSH), have gained attention for securely storing biometric data [19, 36, 38, 43]. LSH projects biometric templates into a hash space, allowing efficient matching while obfuscating the original template. However, recent studies [13, 17, 28, 41] have demonstrated vulnerabilities in LSH-based BTP techniques, as hash codes still contain significant information about the original template. Attackers can exploit the similarity-preserving properties of hash codes to reverse-engineer the biometric data, compromising the privacy and security of the authentication system.

In contrast to LSH-based techniques, homomorphic encryption (HE) offers a promising solution for secure biometric identification in server and cloud environments. HE enables computations on encrypted data without decryption [8, 16], allowing biometric data to remain *fully encrypted* throughout the identification process on the server side. This provides strong privacy protection and mitigates the risks associated with data breaches and unauthorized access. Numerous researchers have developed HE-based privacy-preserving fingerprint identification [14, 26, 46] and face identification [6, 23] techniques. However, the computational overhead introduced by HE operations often leads to slow data matching or searching times, making real-world deployment challenging. Consequently, there is a pressing need for efficient HE-based biometric identification techniques that can achieve practical performance.

Choi et al. [11] recently introduced Blind-Touch, a highly efficient HE-based fingerprint authentication system. However, Blind-Touch employs a fully connected (FC) layer-based metric function, differing from the cosine similarity-based methods typically used in standard biometric authentication architectures, such as

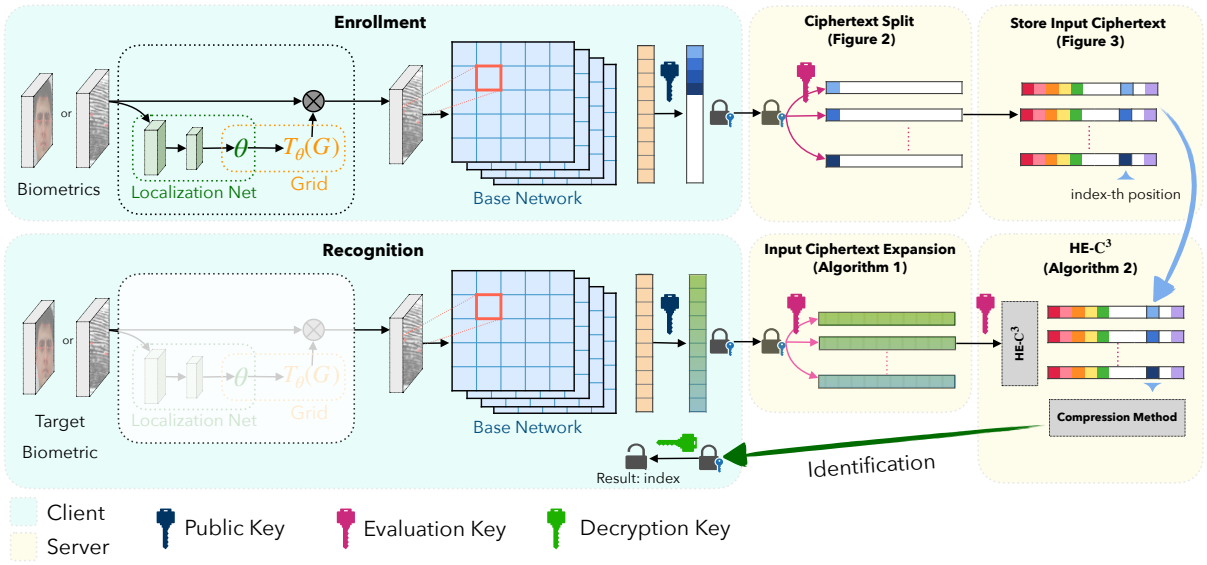

Figure 1: Overview of *Blind-Match*. *Blind-Match* consists of two stages: During the enrollment phase, *Blind-Match* divides and stores the encrypted feature vector into smaller parts. During the recognition phase, *Blind-Match* processes these smaller parts through multiplication and then aggregates the results using our new cosine similarity computation method.

SphereFace [31], CosFace [45], and ArcFace [12]. This FC-layer-based metric usage in Blind-Touch results in a critical limitation in the accuracy of the 1:N (one-to-many) matching task, which is inherently more challenging than the 1:1 matching task.

The 1:N matching task is crucial in real-world scenarios, such as Naver Cloud's FaceSign [3], which enables users to make payments using facial recognition without a credit card. In this service, the system must identify or authenticate an individual by searching through a large database of biometric data without additional identity information. For practical implementation, FaceSign requires the matching task to handle over 5,000 face images within 1 second. This demands a highly accurate and efficient 1:N matching algorithm to ensure a seamless user experience and maintain security. However, our experimental results show that Blind-Touch [11] significantly underperforms in the 1:N matching task, achieving only 59.17% Rank-1 accuracy on the PolyU dataset [29], despite its sufficient speed. This limitation makes the Blind-Touch architecture unsuitable for services like FaceSign, which require both high accuracy and fast processing speed for the 1:N matching task.

To address the challenge of privacy-preserving 1:N biometric matching, we introduce a novel method optimized for HE that computes cosine similarity. The key idea is to partition the feature vector into smaller parts for processing rather than handling the entire vector at once. This technique has led to the development of *Blind-Match*, a robust biometric identification system that achieves Rank-1 accuracies of 99.68% on the PolyU dataset and 99.63% on the LFW dataset, respectively, outperforming the best HE-based method, Blind-Touch [11]. We optimized *Blind-Match*'s computational efficiency by identifying the ideal number of parts to process independently and employing power-of-2 values for template splitting. These techniques allow *Blind-Match* to enable fast 1:N biometric matching, making it suitable for real-world scenarios.

The overall architecture of *Blind-Match* is illustrated in Figure 1. Our key contributions are summarized as follows:

- **Designing a New HE-Optimized Cosine Similarity Algorithm:** We introduce a novel cosine similarity computation method specifically designed to optimize performance in the context of HE. The key idea behind this approach is to divide a biometric feature vector into multiple smaller parts and process each part individually. This technique allows the system to efficiently process large-scale biometric databases, demonstrating its practicality for real-world applications. In our experiments, *Blind-Match* exhibits its impressive speed by processing 6,144 biometric samples in just 0.74 seconds, highlighting its ability to perform a matching task with over 5,000 face images within 1 second, meeting the established criteria for real-world deployment.
- **Demonstrating *Blind-Match*'s Superiority:** The experimental results show that *Blind-Match* significantly outperforms Blind-Touch [11] in fingerprint recognition tasks. On the PolyU fingerprint dataset, *Blind-Match* achieves a 99.55% Rank-1 accuracy, which is 40.38% higher than Blind-Touch. Moreover, across various face datasets, *Blind-Match* consistently achieves high Rank-1 accuracy exceeding 94%, demonstrating its practicality and robustness for real-world biometric identification systems.
- **Releasing Source Code and Preprocessed Datasets:** We make *Blind-Match*'s source code and preprocessed fingerprint dataset publicly available for reproducibility on the GitHub site: https://github.com/hm-choi/blind-match. We believe this can further improve the research in this area and promote and demonstrate practical implementation of our approach in real-world settings.

2 Related Work

Homomorphic Encryption (HE): Homomorphic Encryption (HE) enables computations on encrypted data without decryption,

allowing data owners to entrust third parties with statistical or machine learning operations while maintaining data privacy. Following Gentry's pioneering work in 2009 [16], various HE schemes have been developed, such as the BGV scheme for integer-based arithmetic operations [7] and the CKKS scheme for approximate arithmetic operations over encrypted real and complex numbers [8]. The CKKS scheme has been widely used to preserve privacy in machine learning applications.

The CKKS scheme processes data in batches, with the batch size known as the *number of slots*. The maximum number of allowable multiplications is called the *depth*, which is fixed during key pair generation. If the number of multiplications exceeds the *depth*, the accuracy of the decryption result is not guaranteed. The CKKS scheme allows three operations: Addition (*Add*), Multiplication (*Mul*), and Rotation (*Rot*).

Let N be the *number of slots* and $\mathbf{v_1}, \mathbf{v_2}$ be two real vectors of size N. $C(\mathbf{v_1})$ and $C(\mathbf{v_2})$ denote the ciphertexts of $\mathbf{v_1}$ and $\mathbf{v_2}$ respectively. The above operations are more formally defined as follows:

- Add (P): $Add(C(\mathbf{v_1}), \mathbf{v_2}) = C(\mathbf{v_1} \oplus \mathbf{v_2})$
- Add (C): $Add(C(\mathbf{v_1}), C(\mathbf{v_2})) = C(\mathbf{v_1} \oplus \mathbf{v_2})$
- Mul (P): $Mul(C(\mathbf{v_1}), \mathbf{v_2}) = C(\mathbf{v_1} \otimes \mathbf{v_2})$
- Mul (C): $Mul(C(\mathbf{v_1}), C(\mathbf{v_2})) = C(\mathbf{v_1} \otimes \mathbf{v_2})$
- Rot : $Rot(C(\mathbf{v}), r) = C(v_r, v_{r+1}, ..., v_{N-1}, v_1, ..., v_{r-1})$, where $\mathbf{v} = (v_1, v_2, ..., v_N)$ and r is a non-zero integer.

Here, (P) denotes operations between a ciphertext and a plaintext (or a constant), while (C) denotes operations between two ciphertexts. The notation $\oplus$, and $\otimes$ represents element-wise addition, and multiplication. In this paper, we only use $Add(C)$ for addition, denoted as *Add*. The *Mul* notation represents multiplication with re-linearization, which reduces the ciphertext elements from three to two after multiplication. The *Res* notation represents the rescale operation that reduces noise after multiplication while maintaining ciphertext precision. However, the level l of the ciphertext decreases to $l-1$ after the *Res* operation, indicating the remaining number of allowed multiplications.

In this work, we chose the CKKS scheme for *Blind-Match* due to its efficient support for real number operations, making it the most suitable option for our privacy-preserving machine learning application. In particular, we adopt the *conjugate invariant ring* setting suggested by Kim et al. [24], which supports only real numbers.

2.1 Biometric Recognition with HE

In recent years, there has been a growing interest in developing biometric recognition systems that leverage HE to protect user privacy. There have been multiple efforts to implement fingerprint and face recognition, while preserving privacy using HE.

Face Matching Task. Boddeti et al. [6] propose a fast HE-based 1:1 face matching algorithm, but their architecture is limited to 1:1 matching. As a result, matching time increases proportionally with the number of registered users, leading to poor performance in 1:N matching scenarios (over 10 seconds for a feature vector size of 128 with 1,000 users). To address the 1:N matching task, recent models like SphereFace [32], CosFace [45], ArcFace [12], AdaCos [47], and CurricularFace [22] have effectively employed cosine similarity. Ibarrondo et al. [23] proposed a group testing HE-based face identification using cosine similarity. However, their study used a naive computation method rather than an optimized cosine similarity method for HE, resulting in slow processing speeds.

Fingerprint matching task. Kim et al. [25] developed a HE-based 1:1 matching algorithm utilizing the TFHE library [9]. However, this method requires a considerable amount of time, approximately 166 seconds, for a single matching. Moreover, DeepPrint [14] introduced HE-based 1:N matching, but it requires over 3.4 seconds to search among 5,000 fingerprints, not including the time for encryption and network communication. Additionally, the size of the input ciphertext is about 62 MB. Recently, Blind-Touch [11] has been proposed as a promising solution for real-time HE-based fingerprint 1:1 matching with high accuracy. However, our experiments have demonstrated a significant degradation in Blind-Touch's performance for the 1:N matching task.

To overcome the limitations of existing methods, we introduce *Blind-Match*, which supports standard biometric architectures such as SphereFace, ArcFace, and CosFace by using HE-based cosine similarity computation for high performance in the 1:N matching task for both face and fingerprint recognition scenarios.

3 Overview of *Blind-Match*

The primary objective of *Blind-Match* is to optimize the computation of cosine similarity within the HE context, enabling support for standard biometric identification architectures. By dividing the feature vector into smaller ciphertexts and efficiently processing them on the server side, *Blind-Match* significantly reduces the number of computationally expensive operations, such as rotations, while ensuring accurate matching results.

3.1 Key Generation and Management

In *Blind-Match*, a key set consisting of a public key, evaluation key, and decryption key is generated. The public key is distributed to each client device for encrypting extracted feature vectors from biometric data. The evaluation key is used for *Mul* and *Rot* operations on the server. The decryption key is securely stored on the client device, while only the public key and evaluation key are sent to the server. Modern devices like smartphones and laptops facilitate secure storage through integrated mechanisms, ensuring the privacy and security of the decryption key.

3.2 Enrollment

During enrollment, a client encrypts a user u's feature vector and transmits the resulting ciphertext C_u to the server. The feature vector occupies the initial slots of the ciphertext, with the remaining slots padded with zeros. The *number of slots* in the ciphertext is determined by the degree parameter N. Figure 2(a) shows the structure of C_u.

The server multiplies the received C_u by a **mask**, a set of N_{in} subvectors $\mathbf{mask}_i$, each of length N. The slots of $\mathbf{mask}_i$ are filled with zeros, except for m/N_{in} slots starting from the $i \times m/N_{in}$th index, which are filled with ones, where m represents the size of the feature vector extracted from the CNN extractor (see Figure 2(b)). Without loss of generality, the degree parameter N and feature vector size m are defined as powers of two. Each multiplied ciphertext is then rotated right by $(Index - i + 1) \times m/N_{in}$ slots, where *Index* represents the user's unique identifier. These rotated ciphertexts are added

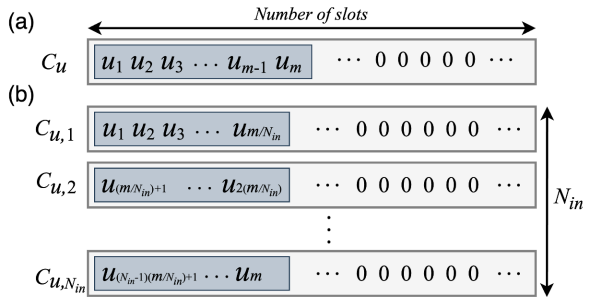

Figure 2: Structure of input ciphertexts.

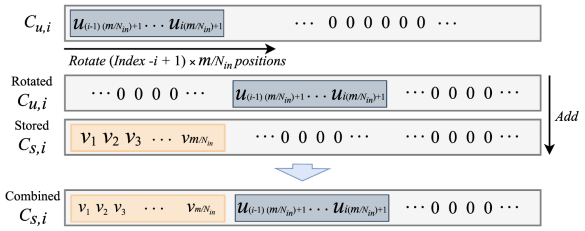

Figure 3: Enrollment of the i-th input ciphertext.

to the previously stored ciphertexts, positioning the user's feature vector at the $Index \times m/N_{in}$ position in the combined ciphertext. The server stores these combined ciphertexts, as described in Figure 3.

3.3 Recognition

During recognition, a client encrypts the user's feature vector and transmits the resulting ciphertext C_u to the server. The server then expands C_u into N_{in} ciphertexts using the Input Ciphertext Expansion algorithm (Algorithm 1). Next, the server computes the cosine similarity with feature vector sub-parts using the HE-Cossim Ciphertext Cloning (HE-C^3) algorithm (Algorithm 2). Finally, the server combines the output ciphertexts into a single ciphertext using compression methods [11] and returns it to the client.

Our cosine similarity computation method divides the input ciphertext into N_{in} expanded ciphertexts (Algorithm 1) and calculates the cosine similarity using the HE-C^3 algorithm (Algorithm 2). Our algorithms are designed to keep the multiplication (*Mul*) operations constant while reducing rotation (*Rot*) operations. As N_{in} increases, the direct computation time for cosine similarity decreases. However, the time to generate N_{in} ciphertexts increases. We have derived and empirically calculated the equation to determine the optimal N_{in} that minimizes the total matching time.

3.3.1 Input Ciphertext Expansion Algorithm. Since generating N_{in} ciphertexts on the client side is computationally expensive and incurs high network costs for transmitting them to the server, we propose an efficient method for expanding the client's single input ciphertext C_u into N_{in} ciphertexts on the server side, as described in Algorithm 1. This approach reduces the computational burden on the client and minimizes the amount of data transmitted over the network.

The **mask** is redefined as a set of real vectors $\mathbf{mask}_i$, where $i \in [0, N_{in})$ with size N, in the following way: for $j \in [0, N)$, $\mathbf{mask}_i[i \cdot m/N_{in} + j] = 1$ if $(i \cdot m/N_{in} + j \bmod m) < m/N_{in}$; otherwise, $\mathbf{mask_i[j]} = 0$.

Algorithm 1 *Input Ciphertext Expansion*

1: **Input:** C_{in} (= C_u) is an input ciphertext, N_{in}, m, $mask = \{mask_i\}_{i=1,2,\ldots,N_{in}}$
2: **Output:** $C_{out} = \{C_{out,i}\}_{i=1,2,\ldots,N_{in}}$
3: Let $PoT(x) = 2^x$.
4: **for** $i = 1$ **to** N_{in} **do**
5: $\quad C_{out,i} = Res(Mul(P)(C_{in}, mask_i))$
6: $\quad$ **for** $j = 1$ **to** $\log N_{in}$ **do**
7: $\quad\quad$ **if** $(i/PoT(j)) \bmod 2 == 0$ **then**
8: $\quad\quad\quad C_{rot,i} = Rot(C_{out,i}, (-1) \cdot PoT(j + \log m - \log N_{in}))$
9: $\quad\quad$ **else**
10: $\quad\quad\quad C_{rot,i} = Rot(C_{out,i}, PoT(j + \log m - \log N_{in}))$
11: $\quad\quad$ **end if**
12: $\quad\quad C_{out,i} = Add(C_{out,i}, C_{rot,i})$
13: $\quad$ **end for**
14: **end for**
15: **return** C_{out}

3.3.2 HE-C^3 Algorithm. The core idea of HE-C^3 is to use N_{in} input ciphertexts instead of a single input ciphertext, as shown in Figure 4.

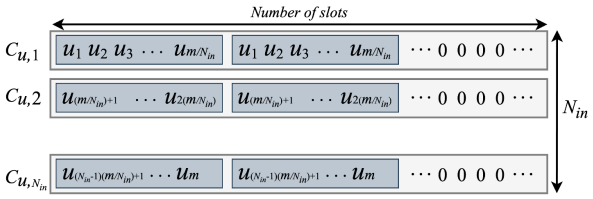

Figure 4: Input ciphertexts for HE-C^3.

The server then executes the matching algorithm described in Algorithm 2, which takes the following inputs: user's input ciphertexts $C_u = \{C_{u,i}\}_{i=1,2,\ldots,N_{in}}$ from Algorithm 1, the server's stored ciphertexts $C_s = \{C_{s,i}\}_{i=1,2,\ldots,N_{in}}$, and the feature vector size m. Algorithm 2 initializes an output ciphertext C_{out} with zero vectors. Then, for each i from 1 to N_{in}, it multiplies the corresponding user and server ciphertexts, rescales the result, and adds it to C_{out}. Finally, it performs a series of rotations and additions on C_{out} based on the values of m and N_{in}, and returns the resulting ciphertext.

This approach significantly reduces the number of computationally expensive rotation operations in HE by effectively reducing the feature vector size from m to m/N_{in} by splitting the feature vectors into smaller parts. This directly reduces the number of *Add* and *Rot* operations required for the inner product operation in line 8 of Algorithm 2.

Although line 4 of Algorithm 2 suggests an increase in *Add*, *Mul*, and *Res* operations proportional to N_{in}, the decreased size of partial feature vectors ($C_{u,i}$ and $C_{s,i}$) by a factor of N_{in} allows the ciphertext to contain N_{in} times more partial feature vectors. As the ciphertext size remains constant, the total number of *Add*, *Mul*, and *Res* operations does not increase with N_{in} from an amortized analysis perspective, effectively reducing the number of *Add* and *Rot* operations without incurring additional computational costs.

Algorithm 2 *HE-Cossim Ciphertext Cloning* (HE-C^3)

Input: $C_u = \{C_{u,i}\}_{i=1,2,\ldots,N_{in}}, C_s = \{C_{s,i}\}_{i=1,2,\ldots,N_{in}}, m$
Output: C_{out}
Let C_{out} be a ciphertext of encryption of zero vectors.
for $i = 1$ **to** N_{in} **do**
  $C_{out} = Add(C_{out}, Res(Mul(C_{u,i}, C_{s,i})))$
end for
for $i = 1$ **to** $(\log m - \log N_{in})$ **do**
  $C_{out} = Add(Rot(C_{out}, 2^{i-1}), C_{out})$
end for
return C_{out}

3.3.3 Compression Method. To minimize client-server communication costs, we employ a *compression method* that combines multiple output ciphertexts into a single ciphertext. As shown in Figure 1, the server applies this method to the expanded ciphertexts after executing Algorithm 2.

3.3.4 Decryption. The server transmits the compressed result ciphertext C_r to the client. The client decrypts C_r using its secret key and identifies the index of the value with the highest similarity in the resulting vector. If this similarity value falls below a predetermined threshold, the client can conclude that user u is not registered with the server.

3.4 Cluster Architecture

Blind-Match uses a scalable cluster architecture with a main server and multiple cluster servers for parallel processing of encrypted ciphertexts. This enables concurrent processing of $(N \cdot N_{in})/m$ feature vectors. For example, with $N = 8,192$, $m = 128$, and $N_{in} = 4$, 256 feature vectors are matched simultaneously, scaling overall matching time by $\lceil R/256 \rceil$ (R being total registered vectors).

During enrollment, encrypted user feature vectors are distributed across expandable clusters. For recognition, the client sends an input ciphertext to the main server, which distributes it to all clusters. Each cluster expands the input and executes Algorithm 2 in parallel. The main server combines the returned outputs using the compression method before sending the final result to the client.

4 Algorithm Analysis

In this section, we find the optimal N_{in} for minimizing the total matching time of *Blind-Match*. The time taken by the Input Ciphertext Expansion algorithm (Algorithm 1), HE-C^3 (Algorithm 2), and compression method are described in Propositions 4.1, 4.2, and 4.3, respectively. Using these propositions, we will demonstrate that the total matching time of *Blind-Match* is minimized when $N_{in} = 4$ with $m = 128$, where $T_{OP,l}$ denotes the time to perform an OP (e.g., *Add*, *Mul*, *Res*, and *Rot*) operation at level l.

Proposition 4.1. *The time it takes to perform Algorithm 1 can be calculated as* $N_{in} \cdot (T_{Mul(P),l} + T_{Res,l}) + \log(N_{in}) \cdot (T_{Res,l-1} + T_{Mul(C),l-1})$.

Proposition 4.2. *The time it takes to perform Algorithm 2 for R feature vectors is* $\lceil (m'/N_{in}) \rceil \cdot (N_{in} \cdot (T_{Mul(C),l} + T_{Res,l} + T_{Add,l-1}) + (\log m' - \log N_{in}) \cdot (T_{Rot,l-1} + T_{Add,l-1}))$, *where* $m' = (m \cdot R)/N$.

Table 1: Comparison of the average execution time (with standard deviation) in milliseconds (*ms*) for the CKKS scheme's operations (*Add*, *Mul*(*C*), *Mul*(*P*), *Rot*, and *Res*) with $N = 8,192$. All experiments were conducted 30 times.

l \op	*Add*	*Mul*(*C*)	*Mul*(*P*)	*Rot*	*Res*
3	0.44 (0.03)	7.87 (0.51)	1.25 (0.13)	7.50 (0.33)	1.30 (0.04)
2	0.33 (0.01)	5.49 (0.34)	0.97 (0.06)	5.33 (0.45)	0.95 (0.02)
1	0.09 (0.01)	3.13 (0.10)	0.76 (0.08)	2.95 (0.16)	0.60 (0.03)

Proposition 4.3. *The time it takes to perform the Compression method is* $\lceil (m'/N_{in}) \rceil \cdot (\log m' - \log N_{in}) \cdot (T_{Mul,l} + T_{Rot,l} + T_{Add,l})$, *where* $m' = (m \cdot R)/N$.

The running time of Algorithm 1 increases with N_{in}, while the running time of Algorithm 2 decreases according to the inverse of N_{in}. Thus, the total matching time is minimized when N_{in} is close to $\sqrt{m'}$. To determine the optimal value of N_{in}, we use the actual execution times of each operation in the formulas. Table 1 presents the average execution time (*ms*) for the CKKS scheme's operations (*Add*, *Mul*(*C*), *Mul*(*P*), *Rot*, and *Res*).

By combining Propositions 4.1, 4.2, and 4.3, and using the values from Table 1, the total matching time $F(N_{in})$ as a function of N_{in} can be written as:

$$
\begin{aligned}
F(N_{in}) = & \underbrace{N_{in} \cdot ((T_{Mul(P),l} + T_{Res,l}) + \log N_{in} \cdot (T_{Res,l-1} + T_{Mul(C),l-1}))}_{\text{by Proposition 4.1}} \\
& + \underbrace{\left\lceil \left(\frac{m'}{N_{in}} \right) \right\rceil \cdot (N_{in} \cdot (T_{Mul(C),l-1} + T_{Res,l-1} + T_{Add,l-2})}_{\text{by Proposition 4.2}} \\
& + \underbrace{(\log m' - \log N_{in}) \cdot (T_{Rot,l-2} + T_{Add,l-2}))}_{\text{by Proposition 4.2}} \\
& + \underbrace{\left\lceil \left(\frac{m'}{N_{in}} \right) \right\rceil \cdot (\log m' - \log N_{in}) \cdot (T_{Mul(P),l-2} + T_{Rot,l-2} + T_{Add,l-2})}_{\text{by Proposition 4.3}}
\end{aligned}
$$

Evaluating the formula for different values of N_{in} with Table 1, we get: $F(2) = 664.7$, $F(4) = 434.84$, $F(8) = 438.64$, $F(16) = 675.6$, and $F(32) = 1320.96$. Therefore, the minimum matching time is obtained when $N_{in} = 4$, with a value of $F(N_{in}) = 434.84$.

5 Experiments

5.1 Experimental Setup

We utilized Lattigo v5 [2], an open-source library for HE, with $N = 8,192$ and $\log PQ \approx 158$ to ensure a 128-bit security level [42]. Our architecture consisted of one client, one server, and three clusters (see Section 3.4). The experiments were conducted using four NAVER Cloud [4] standard-g2 server instances, each equipped with two cores (Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz) and 8GB of DRAM. Each cluster was designed to support 2,048 biometric identifications, resulting in a total capacity of simultaneously storing 6,144 biometric data entries across the three clusters. We employed ResNet-18 (denoted as R18) [20] as the feature extractor for both fingerprints and faces, and the model was trained using the ArcFace loss function to enhance its discriminative power. Since R18 is a highly light CNN architecture, we utilize R18 instead of R50 or larger CNN architectures to reduce the load of the client's device.

5.2 Datasets

We conducted extensive experiments using five face and three fingerprint datasets to evaluate the performance of *Blind-Match*.

5.2.1 Face Datasets. For 1:N face matching training, we primarily used the Glint360k dataset [5], which contains 93K identities and 5.1M images. We followed the preprocessing steps as described in ArcFace [12] to ensure consistency with state-of-the-art methods. To evaluate our model's accuracy, we used several benchmark datasets, including LFW [21], CFP-FP [44], AgeDB [39], and IJB-C [37]. These datasets cover a wide range of variations in facial appearances, poses, and ages, allowing for a comprehensive assessment of our model's performance in real-world scenarios.

5.2.2 Fingerprint Datasets. We evaluated our model's 1:N fingerprint matching performance using four established datasets: FVC2002 and FVC2004 [34], PolyU Cross Sensor [29], and CASIA Version 5.0 [1]. PolyU includes both contact-based and contactless-2D images, while CASIA, the largest public dataset, contains 20,000 images from 4,000 subjects. To improve generalizability, we combined FVC subsets and enhanced image quality in FVC and PolyU through segmentation and centralization.

5.3 Execution Time of *Blind-Match*

We evaluated how efficiently *Blind-Match* can process 6,144 IJB-C samples when HE operations are applied. Table 2 presents the execution time for each step in *Blind-Match* with varying N_{in} values.

Table 2: Mean 1:N matching time (*ms*) over 10 trials for *Blind-Match* with different N_{in} on 6,144 IJB-C samples with the following parameters: $N = 8,192$, $l = 3$, feature vector size = 128. Values in parentheses represent standard deviations.

Operation (OP)	2	4	8	16
Encryption	29.58 (4.32)			
Decryption	24.35 (1.57)			
Inference	129.23 (8.24)			
Matching	650.92 (13.38)	451.67 (9.27)	457.02 (13.92)	652.13 (13.45)
Network	108.52 (7.39)	102.60 (12.71)	100.03 (12.50)	105.55 (19.93)
Total	942.59 (18.59)	737.41 (13.49)	740.20 (12.72)	940.84 (12.73)

Table 2 presents the 1:N matching time using a ResNet-18-based CNN extractor with a 128-dimensional feature vector. The network time measurements confirm that N_{in} minimally impacts *Blind-Match*'s network time. As the CKKS parameter setting remains constant, encryption and decryption times are identical across all N_{in} values. The results show optimal matching time (451.67 ms) and total time (737.41 ms) when $N_{in} = 4$, aligning with the analysis in Section 4. *Blind-Match*'s network time remains consistent at around 110 ms for 6,144 biometric 1:N matchings, outperforming Blind-Touch's 139 ms for 5,000 fingerprint matchings.

Comparison to Conventional HE-based 1:N Biometric Matching Algorithm. We compared the execution time of *Blind-Match*'s 1:N matching algorithm with the conventional cosine similarity-based 1:N matching algorithm (*Base*). Table 3 shows the matching time comparison, revealing that *Blind-Match*'s 1:N matching time is significantly reduced compared to the conventional approach. For a feature vector size of 128, *Blind-Match* achieves a matching time of 451.67 ms, which is nearly 3.5 times faster than *Base*. Furthermore, *Blind-Match* demonstrates its efficiency across different feature vector sizes, with matching times of 136.51 ms and 784.91 ms for feature vector sizes of 16 and 256, respectively.

Table 3: Comparison of mean 1:N matching time (*ms*) over 10 trials comparison between the conventional cosine similarity-based algorithm (*Base*) with a feature vector size of 128 and *Blind-Match* with feature vector sizes of 16, 128, and 256, evaluated on 6,144 samples. Values in parentheses represent the standard deviations.

Operation	*Base* (128)	*Blind-Match* (16)	*Blind-Match* (128)	*Blind-Match* (256)
Matching	1,577.90 (11.43)	136.51 (4.55)	451.67 (9.27)	784.91 (11.35)

5.4 Accuracy of *Blind-Match*

We compared the accuracy of *Blind-Match* with state-of-the-art methods under various conditions.

Accuracy of 1:N Face Matching. Table 4 presents the accuracy of *Blind-Match* with varying feature vector sizes, compared to the state-of-the-art model CosFace with a ResNet-50 (R50) backbone and feature size 512 [45]. For CosFace, we used the results reported in the original paper, which did not include IJB-C. CosFace with R50 and feature size 512 outperforms *Blind-Match* using R18 with smaller feature sizes. While the performance difference is not significant on LFW, *Blind-Match* shows slightly lower accuracy on CFP-FP and AgeDB, even with a feature size of 512. The impact of feature vector size is minimal on LFW, but performance degrades on CFP-FP and AgeDB with feature sizes of 64 or less. For IJB-C, accuracy decreases with feature sizes of 128 or less. On AgeDB and IJB-C, feature vector size significantly impacts accuracy, with a size of 16 resulting in a substantial performance drop. Considering the matching time comparison in Table 3, we recommend a feature vector size of 128 for *Blind-Match* to minimize the negative impact on accuracy and throughput while maintaining competitive performance.

Accuracy of 1:N Fingerprint Matching. We analyzed *Blind-Match*'s performance on the PolyU contactless fingerprint dataset with varying feature vector sizes. The PolyU dataset consists of two sessions, each containing 336 and 160 subjects, respectively, with 6 fingerprint images per subject. The first session was used for training and the second for testing (see Table 5).

Blind-Match demonstrates superior performance across multiple datasets and feature sizes. On the PolyU dataset, it achieves 99.79% Rank-1 accuracy with a 512-feature size and 99.68% with 128 features, significantly outperforming Blind-Touch [11], which achieves only 59.17%. Notably, *Blind-Match*'s performance remains robust even with a 16-feature size, showing only a 0.24% decrease in accuracy compared to the 512-feature model. On the FVC dataset, *Blind-Match* maintains high performance with Rank-1 accuracies of 90.52%, 90.49%, and 90.18% for feature sizes 512, 128, and 16, respectively. The slight performance drop (0.34%) from 512 to 16 features is negligible. The lower overall accuracy on FVC compared to PolyU can be attributed to higher noise levels and the use of

Table 4: Accuracy of *Blind-Match* on face recognition benchmarks for different feature vector sizes, compared to the state-of-the-art CosFace model with ResNet-50 (R50) and feature size 512. Rank-1 accuracy is reported for all datasets except IJB-C, which was not reported [45]. Values in parentheses represent the feature vector size used in each model.

Method	LFW	CFP-FP	AgeDB	IJB-C
CosFace (512), R50 [45]	99.83	99.33	98.55	-
Blind-Match (512)	99.72	95.61	97.28	95.32
Blind-Match (256)	99.58	94.91	97.43	95.36
Blind-Match (128)	99.63	95.54	97.18	94.91
Blind-Match (64)	99.52	94.81	96.10	93.48
Blind-Match (32)	99.43	94.36	95.10	90.86
Blind-Match (16)	99.22	94.00	90.78	83.83

Table 5: Rank-1 accuracy of *Blind-Match* on the PolyU Contactless Fingerprint dataset for different feature sizes.

Feature size	512	256	128	64	32	16
Rank-1 (%)	99.79	99.70	99.68	99.64	99.64	99.55

Table 6: Rank-1 accuracy of *Blind-Match* on the FVC Fingerprint dataset for different feature sizes.

Feature size	512	256	128	64	32	16
Rank-1 (%)	90.52	90.47	90.49	90.43	90.30	90.18

multiple fingerprint readers. For the CASIA dataset, *Blind-Match* achieves impressive Rank-1 accuracies of 99.97% and 99.87% with feature sizes 16 and 128, respectively.

Given the consistent high performance across feature sizes and the small-resolution, black-and-white nature of the PolyU, FVC, and CASIA datasets, we recommend using a 16-feature size for *Blind-Match* in 1:N fingerprint matching tasks. This configuration offers an optimal balance between computational efficiency and accuracy. Table 6 provides detailed results for the FVC dataset.

Accuracy of 1:1 Fingerprint Matching. We compared AUC and EER scores for the PolyU dataset with state-of-the-art 1:1 fingerprint-matching architectures. The dataset was split into train and test sets using the configuration in [15], with 3,000 genuine pairs and 19,900 imposter pairs in the test set. All models except *Blind-Match* (feature size 128) and Blind-Touch [11] used plaintext data. Table 7 shows that *Blind-Match*'s 1:1 matching accuracy, in terms of AUC and EER scores, is close to that of state-of-the-art architectures [35]. When trained on 1:N matching, *Blind-Match* achieves an AUC score of 98.55%, only 0.78% lower than the best plaintext model, ContactlessMinuNet [48]. However, the EER score of 5.9% is slightly higher due to *Blind-Match*'s optimization for 1:N matching. This optimization prioritizes overall performance across thresholds rather than optimizing for a specific threshold, which explains the high AUC alongside the higher EER.

Table 7: AUC and EER of *Blind-Match* and state-of-the-art methods on the PolyU dataset for 1:1 fingerprint matching.

Method	AUC (%)	EER (%)
MNIST mindtct [27]	58.91	36.85
MinutiaeNet [40]	93.03	13.35
VeriFinger (paid software)	98.16	2.99
ContactlessMinuNet [48]	99.33	1.94
MinNet [15]	99.25	1.90
Blind-Touch [11]	97.50	2.50
Blind-Match (128)	98.55	5.90

6 Security Analysis

Blind-Match considers an honest-but-curious server adversary model, where the server follows the protocol but can observe encrypted feature vector ciphertexts. The system's security relies on the CKKS scheme [8], a fully HE scheme that provides semantic security against chosen-plaintext attacks (IND-CPA). The security of the CKKS scheme is based on the hardness of the decisional ring learning with errors problem [33], ensuring that an adversary cannot distinguish between encryptions of different messages.

In *Blind-Match*, the client encrypts feature vectors using the CKKS scheme before sending them to the server, which performs matching on the encrypted vectors without decrypting them. Due to the IND-CPA security of the CKKS scheme, the server learns nothing about the plaintext feature vectors from the ciphertexts except for their length. The feature extractor always produces ciphertexts of consistent length, ensuring no additional information can be inferred. More formally, the security of *Blind-Match* can be analyzed using the simulation paradigm [30], where an ideal functionality receives feature vectors, performs matching, and returns the result to the client. Any attack on *Blind-Match* can be simulated as an attack on this ideal functionality, implying the system's security against honest-but-curious adversaries. However, HE only provides confidentiality and does not protect against integrity issues or denial-of-service attacks.

7 Deployment Plan and Conclusion

Our proof-of-concept implementation demonstrated *Blind-Match*'s exceptional efficiency, matching over 5,000 face images and performing 6,144 identifications in just 0.74 seconds across multiple biometric modalities. This establishes *Blind-Match* as the first real-time homomorphic encryption-based architecture for 1:N biometric matching with practical applicability.

Future work will deploy *Blind-Match* in real-world scenarios, expanding its functionality to Android devices via a cloud-based matching server. To evaluate performance and scalability, we plan to implement *Blind-Match* in a large research institution's entrance system, serving over 4,000 employees.

Acknowledgments

This work was partly supported by grants from the IITP (RS-2022-II220688, RS-2021-II212068, No.2022-0-01199, RS-2023-00229400, RS-2019-II190421, and RS-2024-00439762).

References

[1] 2014. CASIA-FingerprintV5 Database. Online: http://www.idealtest.org/. accessed 20-Jan-2024.

[2] 2023. Lattigo v5. Online: https://github.com/tuneinsight/lattigo. EPFL-LDS, Tune Insight SA, accessed 18-May-2024.

[3] 2024. FaceSign. Online: https://m.pulsenews.co.kr/view.php?sc=30800025&year=2024&no=179846. accessed 10-May-2024.

[4] 2024. NAVERCloud. Online: https://www.ncloud.com/product/compute/server. accessed 18-May-2024.

[5] Xiang An, Jiankang Deng, Jia Guo, Ziyong Feng, XuHan Zhu, Jing Yang, and Tongliang Liu. 2022. Killing Two Birds With One Stone: Efficient and Robust Training of Face Recognition CNNs by Partial FC. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4042–4051.

[6] Vishnu Naresh Boddeti. 2018. Secure face matching using fully homomorphic encryption. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*. IEEE, 1–10.

[7] 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.

[8] 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*. 409–437.

[9] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. August 2016. TFHE: Fast Fully Homomorphic Encryption Library. https://tfhe.github.io/tfhe/.

[10] Geumhwan Cho, Jun Ho Huh, Soolin Kim, Junsung Cho, Heesung Park, Yenah Lee, Konstantin Beznosov, and Hyoungshick Kim. 2020. On the security and usability implications of providing multiple authentication choices on smartphones: The more, the better? *ACM Transactions on Privacy and Security (TOPS)* 23, 4 (2020), 1–32.

[11] Hyunmin Choi, Simon S Woo, and Hyoungshick Kim. 2024. Blind-Touch: Homomorphic Encryption-Based Distributed Neural Network Inference for Privacy-Preserving Fingerprint Authentication. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 21976–21985.

[12] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. Arcface: Additive angular margin loss for deep face recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4690–4699.

[13] Xingbo Dong, Zhe Jin, and Andrew Teoh Beng Jin. 2019. A genetic algorithm enabled similarity-based attack on cancellable biometrics. In *IEEE 10th International Conference on Biometrics Theory, Applications and Systems (BTAS)*.

[14] Joshua J Engelsma, Kai Cao, and Anil K Jain. 2019. Learning a fixed-length fingerprint representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 6 (2019), 1981–1997.

[15] Yulin Feng and Ajay Kumar. 2023. Detecting locally, patching globally: An end-to-end framework for high speed and accurate detection of fingerprint minutiae. *IEEE Transactions on Information Forensics and Security* 18 (2023), 1720–1733.

[16] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the ACM symposium on Theory of computing*. 169–178.

[17] 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 (TIFS)* (2020).

[18] Stephanie Gootman. 2016. OPM hack: The most dangerous threat to the federal government today. *Journal of Applied Security Research* (2016), 517–525.

[19] 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 (TIFS)* (2022).

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs.CV]

[21] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.* Technical Report 07-49. University of Massachusetts.

[22] Yuge Huang, Yuhan Wang, Ying Tai, Xiaoming Liu, Pengcheng Shen, Shaoxin Li, Jilin Li, and Feiyue Huang. 2020. CurricularFace: Adaptive Curriculum Learning Loss for Deep Face Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

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

[24] Duhyeong Kim and Yongsoo Song. 2019. Approximate homomorphic encryption over the conjugate-invariant ring. In *International Conference on Information Security and Cryptology*. Springer, 85–102.

[25] Taeyun Kim, Yongwoo Oh, and Hyoungshick Kim. 2020. Efficient privacy-preserving fingerprint-based authentication system using fully homomorphic encryption. *Security and Communication Networks* 2020 (2020), 1–11.

[26] Taeyun Kim, Yongwoo Oh, Hyoungshick Kim, and José María de Fuentes. 2020. Efficient Privacy-Preserving Fingerprint-Based Authentication System Using Fully Homomorphic Encryption. *Security and Communication Networks* (Jan 2020), 11 pages. https://doi.org/10.1155/2020/4195852

[27] Kenneth Ko. 2007. User's guide to nist biometric image software (nbis). (2007).

[28] 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 (TIFS)* (2021).

[29] Chenhao Lin and Ajay Kumar. 2018. Matching contactless and contact-based conventional fingerprint images for biometrics identification. *IEEE Transactions on Image Processing* 27, 4 (2018), 2008–2021.

[30] Yehuda Lindell. 2017. How to simulate it–a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich* (2017), 277–346.

[31] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2017. Sphereface: Deep hypersphere embedding for face recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 212–220.

[32] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2017. SphereFace: Deep Hypersphere Embedding for Face Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[33] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2013. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)* (2013), 1–35.

[34] Dario Maio, Davide Maltoni, Raffaele Cappelli, James L Wayman, and Anil K Jain. 2002. FVC2002: Second fingerprint verification competition. In *International conference on pattern recognition*, Vol. 3. IEEE, 811–814.

[35] Davide Maltoni, Dario Maio, Anil K Jain, and Jianjiang Feng. 2022. *Handbook of Fingerprint Recognition.* Springer Nature.

[36] Manisha and Nitin Kumar. 2020. Cancelable biometrics: a comprehensive survey. *Artificial Intelligence Review* (2020).

[37] Brianna Maze, Jocelyn Adams, James A. Duncan, Nathan Kalka, Tim Miller, Charles Otto, Anil K. Jain, W. Tyler Niggel, Janet Anderson, Jordan Cheney, and Patrick Grother. 2018. IARPA Janus Benchmark - C: Face Dataset and Protocol. In *International Conference on Biometrics (ICB)*. 158–165. https://doi.org/10.1109/ICB2018.2018.00033

[38] Blaž Meden, Peter Rot, Philipp Terhörst, Naser Damer, Arjan Kuijper, Walter J Scheirer, Arun Ross, Peter Peer, and Vitomir Štruc. 2021. Privacy–enhancing face biometrics: A comprehensive survey. *IEEE Transactions on Information Forensics and Security (TIFS)* (2021).

[39] Stylianos Moschoglou, Athanasios Papaioannou, Christos Sagonas, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. 2017. AgeDB: The First Manually Collected, In-the-Wild Age Database. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

[40] Dinh-Luan Nguyen, Kai Cao, and Anil K Jain. 2018. Robust minutiae extractor: Integrating deep networks and fingerprint domain knowledge. In *2018 International Conference on Biometrics (ICB)*. IEEE, 9–16.

[41] Seunghun Paik, Sunpill Kim, and Jae Hong Seo. 2023. Security Analysis on Locality-Sensitive Hashing-Based Biometric Template Protection Schemes. (2023).

[42] Yogachandran Rahulamathavan. 2022. Privacy-preserving Similarity Calculation of Speaker Features Using Fully Homomorphic Encryption. arXiv:2202.07994 [cs.CR]

[43] Mulagala Sandhya and Munaga VNK Prasad. 2017. Biometric template protection: A systematic literature review of approaches and modalities. *Biometric Security and Privacy: Opportunities & Challenges in The Big Data Era* (2017).

[44] Soumyadip Sengupta, Jun-Cheng Chen, Carlos Castillo, Vishal M. Patel, Rama Chellappa, and David W. Jacobs. 2016. Frontal to profile face verification in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 1–9. https://doi.org/10.1109/WACV.2016.7477558

[45] 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. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5265–5274.

[46] Wencheng Yang, Song Wang, Kan Yu, James Jin Kang, and Michael N Johnstone. 2020. Secure fingerprint authentication with homomorphic encryption. In *Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 1–6.

[47] Xiao Zhang, Rui Zhao, Yu Qiao, Xiaogang Wang, and Hongsheng Li. 2019. AdaCos: Adaptively Scaling Cosine Logits for Effectively Learning Deep Face Representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[48] Zhao Zhang, Shuxin Liu, and Manhua Liu. 2021. A multi-task fully deep convolutional neural network for contactless fingerprint minutiae extraction. *Pattern Recognition* 120 (2021), 108189.