

Article

A Secure Face Verification Scheme Based on Fully Homomorphic Encryption with Anonymity

Xingchen Wang ^{1,*}  and Peng Li ^{1,2} 

¹ Department of Mathematics and Physics, North China Electric Power University, Baoding 071003, China; lphit@163.com

² Hebei Key Laboratory of Physics and Energy Technology, North China Electric Power University, Baoding 071003, China

* Correspondence: wxc000604@163.com

Abstract: With the widespread adoption of cloud computing, the face verification process often requires the client to upload the face to an untrusted cloud server to obtain the verification results. Privacy leakage issues may arise if the client's private information is not protected. This paper proposes a secure and anonymous face verification scheme using fully homomorphic encryption technology and SealPIR. Our scheme is a three-party solution that requires a third-party server trusted by the client. This scheme not only prevents the client's facial data from being obtained by untrusted data servers but also prevents the data server from learning the index corresponding to the face that the client wants to verify. In a single-face verification process, the client only needs to perform one upload operation and one download operation, with a communication volume of 264 KB. We can complete a privacy-protected anonymous face verification process in 84.91 ms.

Keywords: face verification; anonymity; fully homomorphic encryption; private information retrieval; privacy-protection



Citation: Wang, X.; Li, P. A Secure Face Verification Scheme Based on Fully Homomorphic Encryption with Anonymity. *Information* **2024**, *15*, 129. <https://doi.org/10.3390/info15030129>

Academic Editor: Aneta Ponsiszewska-Maranda

Received: 27 January 2024

Revised: 20 February 2024

Accepted: 22 February 2024

Published: 24 February 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Face recognition technology is a biometric recognition method that extracts facial features for the purpose of identity recognition. Widely applied in security systems, financial institutions, and mobile payment applications, face recognition can be categorized based on its functionality into face verification and face identification [1]. Face verification is a one-to-one face authentication performed by the server on the face uploaded by the client, which is used to determine whether these two faces belong to the same individual. Consequently, the client is required to upload the corresponding index of the face to be verified. Face identification is the one-to-many face matching performed by the server on the face uploaded by the client, which is used to determine the identity information corresponding to this face.

In today's rapidly advancing era of cloud computing, privacy concerns arising from face verification on the cloud have garnered widespread attention. Clients typically need to request a cloud server with a database for the face verification process. In such scenarios, conventional secure face verification schemes often employ cryptographic techniques to encrypt the client's facial information. The encrypted facial data and corresponding indices are then uploaded to a cloud server for decryption, followed by the face verification process. This cloud server may be untrusted. Therefore, addressing the crucial issue of enabling servers to conduct face verification processes without the need for decryption, or even without the server being aware of which specific face the client intends to verify, becomes a significant focus of attention. Because, under typical circumstances, the index information uploaded by the client may expose the client's identity.

Due to fully homomorphic encryption technology, which ensures homomorphic properties between ciphertext and corresponding plaintext, supporting computations on encrypted data, it can be applied to the encrypted computation process of cloud-based face verification [2]. Private information retrieval (PIR) technology allows clients to retrieve specific entries from an untrusted cloud server's database without revealing which particular entry the client is interested in [3]. Therefore, PIR can be utilized to address the query-face process in cloud-based face verification, making it applicable to anonymous face verification schemes.

The main contributions of this paper are as follows:

- Firstly, we propose a three-party secure and anonymous face verification scheme involving the client, data server, and authentication server in this paper. The authentication server requires the trust of the client, while the data server is untrusted by the client, and it is essential to ensure non-collusion among the servers. Our proposed scheme can be applicable in scenarios where a group of clients belonging to the same company, application, or organization require access to face verification services provided by a server (referred to as the data server in our paper). These clients may have limited communication and computing capabilities, are not always online or active, and do not wish to bear the risk of private key exposure. This group of clients trusts a powerful server within their organization (referred to as the authentication server in our paper), and they expect the authentication server to assist in handling partial communication and computation tasks on their behalf and obtain verification results from the authentication server. These scenarios are common in practice.
- Secondly, this paper employs fully homomorphic encryption technology, along with SealPIR [4] based on fully homomorphic encryption. The client uploads encrypted facial feature information and index information to the data server. With the assistance of the authentication server, the data server accomplishes the homomorphically encrypted PIR process and the encrypted facial similarity calculation process. Subsequently, the data server sends the encrypted facial similarity calculation results along with a threshold to the authentication server. The authentication server compares the results and obtains the verification outcome, which is then transmitted to the client. Throughout the entire process, the client is required to engage in only two communications.
- Finally, throughout the entire face verification process, the honest-but-curious (semi-honest) data server cannot obtain any content related to the client's facial feature information and index information. Aside from the verification result, the client cannot access any additional information. In our proposed scheme, the private key is generated by the authentication server, thereby avoiding the risk of private key exposure for the client. A single secure and anonymous face verification process requires only 84.91 ms, with the client's communication volume limited to only 264 KB.

The remaining sections of this work are organized as follows. Section 2 provides a review of some previous relevant literature. Section 3 introduces the preliminaries required for our proposed scheme. Section 4 details the process and security analysis of our proposed scheme. Section 5 presents the evaluations. Section 6 concludes this work.

2. Related Works

2.1. Face Verification Scheme

The face verification schemes employ a client-server model, and depending on the number of parties, they can be categorized into two-party and three-party schemes.

In 2013, Troncoso-Pastoriza et al. [5] proposed a privacy-preserving non-interactive face verification scheme utilizing Gabor features, with significant computational overhead. In 2016, Im et al. [6] transformed an additive homomorphic encryption scheme into a scheme capable of evaluating degree-2 functions, presenting a privacy-preserving biometric verification protocol based on a quadratic homomorphic encryption scheme. In 2017, Abidin [7] introduced a three-party face verification scheme by combining homomorphic

encryption and verifiable computation. In the same year, Ma et al. [8] proposed a three-party face verification scheme based on the Paillier cryptosystem. However, the private key is stored on the server, potentially raising concerns about privacy information leakage. In 2018, Boddeti [9] presented a face verification scheme based on fully homomorphic encryption utilizing batching. In the same year, Lin et al. [10] introduced a three-party scheme using the Paillier cryptosystem and garbled circuits, with the drawback that the server can obtain verification results.

In 2020, Kolberg et al. [11] proposed a three-party face verification scheme, comparing CKKS, BFV, and NTRU [12] homomorphic encryption schemes. However, the authentication server could directly decrypt the distance of facial similarities, potentially inferring certain information about the data server model. In the same year, Im et al. [13] transformed a LinearHE into a semantically secure quadratic homomorphic encryption scheme, presenting a two-party solution with relatively high client-server interactions. Huang and Wang [14], in 2021, proposed a three-party face verification scheme using the BGV encryption scheme. In 2022, Sun et al. [15] utilized Hamming distance to propose a two-party scheme based on the BFV encryption scheme. However, the client could decrypt the distance and threshold, an undesirable scenario for the server.

However, the schemes above do not address the issue of anonymity, where the server can determine which specific face the user intends to verify. These approaches typically involve the client directly uploading index information to the server, which, in practical applications, often becomes a notable privacy concern.

2.2. Private Information Retrieval (PIR)

In 1995, Chor et al. [3] introduced the concept of private information retrieval (PIR), which addresses the scenario where a client wishes to retrieve an item from an untrusted server's database without the server knowing which specific item the client is retrieving. PIR can be categorized based on technical routes into information-theoretic PIR (IT-PIR) and computational PIR (CPIR).

For IT-PIR [3,16–24], the database is replicated across multiple non-colluding servers. The client sends queries to each server and integrates responses locally. In IT-PIR, the computation cost at the servers is relatively inexpensive compared to CPIR. The security of IT-PIR is based on information-theoretic, avoiding cryptographic hardness assumptions and resisting attacks from adversaries with unlimited computational power. However, the requirement for multiple servers to possess the same database and remain non-colluding makes practical deployment of IT-PIR challenging.

For CPIR, the database can exist solely on a single server without relying on multiple non-colluding servers. The security of CPIR is based on cryptographic hardness assumptions. The computational cost of CPIR is often higher than IT-PIR because it involves expensive encryption operations. However, deployment in practical applications is relatively straightforward. The SealPIR scheme, which we are focusing on, falls under the category of CPIR.

In 1997, Kushilevitz and Ostrovsky [25] proposed the first single-server CPIR scheme. This scheme represented N data items in the server's database as d -dimensional hypercubes. Cachin et al. [26], Chang [27], Gentry, and Ramzan [28] conducted subsequent research along this line. In 2007, Sion and Carbunar [29] noted that the query speeds of these protocols in practical applications were slower than downloading the entire database. Additionally, these CPIR schemes based on traditional cryptographic systems were deemed incapable of withstanding quantum attacks. In 2016, Melchor et al. [30] introduced a PIR scheme, XPIR, utilizing a lattice-based cryptographic system resilient to quantum attacks. This scheme aimed to reduce the computational cost of the server. Subsequently, in 2018, Angel et al. [4] proposed a PIR scheme called SealPIR, utilizing the BFV encryption scheme to reduce client communication overhead. In 2021, Mughees et al. [31] presented OnionPIR, a PIR scheme capable of reducing server response sizes. In the same year, Ahmad et al. [32] designed FastPIR, a PIR scheme that decreases server computational costs by increasing

client request sizes. In 2023, Mughees and Ren [33] introduced BatchPIR, a PIR scheme based on the BFV encryption scheme that allows for batch processing.

3. Preliminaries

3.1. Facenet

In a face verification system, it is common to extract facial feature vectors from facial data. The verification result is obtained by comparing the extracted facial feature vector with the facial feature vector templates stored in the database. Deep learning models can extract facial features from raw facial images and have widespread application in the face recognition field.

Facenet [34] utilizes the deep learning model. It was proposed by Google in 2015 and employs a Convolutional Neural Network (CNN) and triplet loss. The CNN is capable of extracting facial feature vectors in the Euclidean space from facial images. It transforms face similarity into the Euclidean distance between their corresponding facial feature vectors. The verification result is obtained by comparing the magnitude of the Euclidean distance with a predefined threshold. Assuming $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ are two face feature vectors, the distance (Euclidean distance) between them is defined as $dist = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$.

We opt for the MobilenetV1 [35] model as a substitute for the Inception-ResNetV1, serving as the backbone feature extraction network in Facenet. MobilenetV1, designed for mobile applications, represents a lightweight deep neural network capable of reducing model parameters and computational complexity. MobilenetV1 utilizes depthwise separable convolution blocks composed of 3×3 depthwise convolutions and 1×1 pointwise convolutions for feature extraction. The primary purpose of depthwise convolution is feature extraction, while pointwise convolution is used to adjust the number of channels. This structure allows depthwise separable convolution blocks to use fewer parameters to replace regular 3×3 convolutions.

Figure 1 illustrates the training phase of the Facenet model. In the prediction phase of the Facenet model, firstly, a feature layer is obtained through the backbone feature extraction network. This feature layer is then flattened and passed through a fully connected layer with 128 neurons, resulting in a 128-dimensional feature vector. At this point, the 128-dimensional feature vector serves as a representation of the input face image. Subsequently, the feature vector undergoes L2 normalization, where each vector element is divided by its L2 norm. L2 normalization ensures that the components of the feature vector are maintained within the same magnitude. We use Triplet Loss and Cross-Entropy Loss to construct the classifier as the loss functions. Triplet Loss is employed to increase the Euclidean distance between feature vectors of different individuals' faces and decrease the Euclidean distance between feature vectors of the same individual's faces. Cross-Entropy Loss is utilized for face classification and assists in the convergence of Triplet Loss. In our work, we utilize Facenet to extract facial feature vectors, which have been L2 normalized. These feature vectors are then encrypted using a fully homomorphic encryption scheme for ciphertext computations.

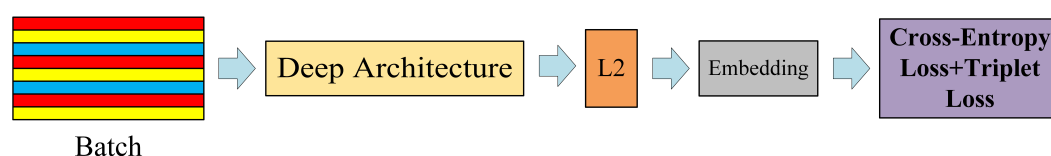


Figure 1. Facenet model training phase.

3.2. Fully Homomorphic Encryption

In 2009, Gentry [36] constructed the first fully homomorphic encryption scheme based on ideal lattices. Subsequently, a cohort of outstanding schemes emerged, including BGV [37], BFV [38], CKKS [39], TFHE [40], etc. BGV, BFV, and CKKS are constructed based on arithmetic circuits, while TFHE relies on binary gates. Fully homomorphic encryption schemes are based on the hardness problems of learning with errors (LWE) and ring learning with errors (RLWE) and are one of the ways to realize post-quantum cryptography that is resistant to quantum attacks. These schemes exhibit an excellent structure, maintaining homomorphism for addition and multiplication operations between ciphertexts and their corresponding plaintexts, allowing the computations on ciphertexts. However, achieving arbitrary ciphertext addition and multiplication levels requires an expensive bootstrapping operation to refresh ciphertext noise. The schemes BGV, BFV, and CKKS employ modulo-switching and key-switching operations to control noise growth, reducing reliance on bootstrapping to some extent. With a level structure, these schemes can support a finite number of ciphertext multiplication operations. This work utilizes the BFV encryption scheme from the fully homomorphic encryption library SEAL [41].

3.2.1. BFV

The BFV encryption scheme supports the encryption of integers. The BFV encryption scheme primarily consists of six algorithms: KEYGEN, ENCRYPT, HOMORADD, HOMOMULT, RELINEARIZATION, and DECRYPT. Algorithms 1–6 show the pseudocode for the six algorithms of the BFV encryption scheme. Table 1 provides the notation used in Algorithms 1–6. Algorithm 1 is used to generate keys. Algorithm 2 first maps the message m from the message space to the plaintext space R_t , and the algorithm is used to encrypt data. Algorithms 3 and Algorithm 4 represent performing homomorphic addition and homomorphic multiplication on ciphertexts ct_1 and ct_2 , respectively. Algorithm 5 is used to reduce the dimension of the ciphertext ct_{prod} from R_q^3 to R_q^2 . Algorithm 6 is used to decrypt ciphertexts.

Table 1. Notation used in Algorithms 1–6.

Notation	Description
$\mathbb{Z}$	Finite field
$\mathbb{Z}[x]$	Polynomials with coefficients in the finite field
n	Power of two
$R = \mathbb{Z}[x]/(x^n + 1)$	Polynomial ring
$\mathbb{Z}_t[x]$	Polynomials $\mathbb{Z}[x]$ with coefficients modulo t
$R_t = \mathbb{Z}_t[x]/(x^n + 1)$	Plaintext space
R_q^2	Ciphertext space
λ	Security level parameter
Δ	$\lfloor q/t \rfloor$
χ	A B-bounded probability distribution over R_q
sk	Private key
pk	Public key
a	Uniformly randomly chosen
e	Noise
T	Bit-decomposition modulus
a_i	Uniformly randomly chosen
e_i	Noise
ct	Ciphertext
$\otimes$	Tensor product

Algorithm 1 KEYGEN**Input:** λ, q, n **Output:** $sk, pk, rk \in R_q^2$

- 1: $sk = (1, s), s \in \chi$
- 2: $pk = (pk_0, pk_1) = (b, a) = ([-(a \cdot s + e)]_q, a), a \in R_q, e \in \chi$
- 3: $rk = (rk_0, rk_1) = ([-(a_i \cdot s + e_i) + T^i \cdot s^2]_q, a_i), i \in [1, \dots, l], a_i \in R_q, e_i \in \chi$

Algorithm 2 ENCRYPT**Input:** m, pk **Output:** $ct \in R_q^2$

- 1: message space $\rightarrow R_t$
- 2: $ct[0] = [pk_0 \cdot u + e_1 + \Delta \cdot m]_q, ct[1] = [pk_1 \cdot u + e_2]_q; u, e_1, e_2 \in \chi$
- 3: $ct = (ct[0], ct[1])$

Algorithm 3 HOMORADD**Input:** ct_1, ct_2 **Output:** $ct_{sum} \in R_q^2$

- 1: $ct_{sum} = ct_1 + ct_2 = ([ct_1[0] + ct_2[0]]_q, [ct_1[1] + ct_2[1]]_q)$

Algorithm 4 HOMORMULT**Input:** ct_1, ct_2 **Output:** $ct_{prod} \in R_q^3$

- 1: $ct_{prod}[0] = \left[\left[\frac{t \cdot (ct_1[0] \cdot ct_2[0])}{q} \right] \right]_q$
- 2: $ct_{prod}[1] = \left[\left[\frac{t \cdot (ct_1[0] \cdot ct_2[1] + ct_1[1] \cdot ct_2[0])}{q} \right] \right]_q$
- 3: $ct_{prod}[2] = \left[\left[\frac{t \cdot (ct_1[1] \cdot ct_2[1])}{q} \right] \right]_q$
- 4: $ct_{prod} = ct_1 \otimes ct_2 = (ct_{prod}[0], ct_{prod}[1], ct_{prod}[2])$

Algorithm 5 RELINEARIZATION**Input:** ct_{prod}, rk **Output:** $ct_t \in R_q^2$

- 1: $c_0 = \sum_{i=0}^l rk_0[i] \cdot ct_{prod}[2]^{(i)}$
- 2: $c_1 = \sum_{i=0}^l rk_1[i] \cdot ct_{prod}[2]^{(i)}$
- 3: $ct_t[0] = [ct_{prod}[0] + c_0]_q$
- 4: $ct_t[1] = [ct_{prod}[1] + c_1]_q$
- 5: $ct_t = (ct_t[0], ct_t[1])$

Algorithm 6 DECRYPT**Input:** ct, sk **Output:** $m \in R_t$

- 1: $\langle ct, sk \rangle = ct[0] + ct[1] \cdot s$
- 2: $m = \left[\left[\frac{t \cdot \langle ct, sk \rangle}{q} \right] \right]_t$

3.2.2. SIMD

The Single Instruction Multiple Data (SIMD) technique, introduced by Smart and Vercauteren [42] in 2014, addresses batch processing issues [43]. Based on the Chinese Remainder Theorem (CRT), SIMD decomposes modular operations on large prime numbers into a series of modular operations on smaller primes. Due to the relative independence of these operations on smaller primes, they can be computed in parallel. Subsequently, the results are reconstructed using CRT to restore the original large prime. The introduction of SIMD significantly improves computational efficiency, allowing the simultaneous processing of multiple slots for identical operations in a single operation. Through SIMD, for a BFV encryption scheme with a polynomial degree of n , BFV can encode up to n slots.

Assuming $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$ are two message vectors, applying SIMD technology to perform homomorphic addition on the ciphertexts of a and b involves encrypting the result of the element-wise sum of a and b . Similarly, homomorphic multiplication on the ciphertexts of a and b encrypts the result of the element-wise product of a and b . Algorithms 7 and 8 illustrate the pseudocode for these processes. However, SIMD has a drawback: the inability to individually access and process individual elements in an encrypted vector.

Algorithm 7 HOMORADD based on SIMD technique

Input: ENCRYPT($a = (a_1, \dots, a_n)$, pk), ENCRYPT($b = (b_1, \dots, b_n)$, pk)

Output: ct_{a+b}

1: $ct_{a+b} = \text{ENCRYPT}((a_1 + b_1, \dots, a_n + b_n), pk)$

Algorithm 8 HOMORMULT based on SIMD technique

Input: ENCRYPT($a = (a_1, \dots, a_n)$, pk), ENCRYPT($b = (b_1, \dots, b_n)$, pk)

Output: $ct_{a \times b}$

1: $ct_{a \times b} = \text{ENCRYPT}((a_1 \times b_1, \dots, a_n \times b_n), pk)$

3.2.3. Rotation

Due to SIMD technology, direct operations like summing all elements in an encrypted vector are impossible. The rotation operation introduced by Gentry et al. [44] can address this issue. The rotation operation requires Galois keys, which can be generated by the KEYGEN algorithm, constructed similarly to the relinearization key rk . Different lengths of rotations correspond to different Galois keys. For an encrypted vector of length n , by performing $\log_2 n$ rotation operations and homomorphic addition operations, the sum of all elements of this vector can be obtained in the first slot of the encrypted vector. Therefore, KEYGEN typically generates $\log_2 n$ Galois keys. Note that the rotation operation rotates to the left by default. The pseudocode for rotation is presented in Algorithm 9.

Algorithm 9 ROTATE

Input: ENCRYPT($(x_0, x_1, \dots, x_{n-1})$, pk), k , gk

Output: ct_{rot}

1: $ct_{rot} = \text{ENCRYPT}((x_k, \dots, x_{n-1}, x_0, \dots, x_k - 1), pk)$

3.3. SealPIR

SealPIR is based on the BFV encryption scheme and belongs to CPIR. SealPIR has two main contributions. Firstly, it encodes the query of the client into polynomial x^i , where i represents the index. The polynomial is then encrypted and sent to the server, reducing the communication overhead of the client and compressing the query. Secondly, SealPIR introduces a novel data encoding using probabilistic batch codes (PBC) to construct a multi-query PIR scheme, allowing the server to amortize its computation cost when processing

a batch of requests from the same client. Since our work focuses on a single-query PIR scheme, the emphasis is on the former.

Due to the polynomial degree being at most n , a single query ciphertext can only query a database of size n . To increase the query database size, SealPIR represents a database with N entries as a d -dimensional hypercube, where the client can complete a PIR query by sending d index ciphertexts, typically with d set to 2. Upon receiving these d ciphertexts, the server can use an algorithm called EXPAND to extend these ciphertexts, turning one ciphertext into $\sqrt[d]{N}$ ciphertexts. After applying the EXPAND algorithm, a ciphertext with index i results in a batch of ciphertexts where the i -th ciphertext is encryption of 1, and the others are encryptions of 0. In this way, the outcome yields $d \sqrt[d]{N}$ ciphertexts.

In the case of $d = 2$, for a database of size N , the server preprocesses the database into an $\sqrt{N} \times \sqrt{N}$ matrix M , where each element represents an entry in the database. M is a plaintext. Assuming the user wants to query the element $M[r, c]$, the client sends two request ciphertexts, $q_{row} = \text{ENCRYPT}(x^r, pk)$ and $q_{col} = \text{ENCRYPT}(x^c, pk)$. The server uses the EXPAND algorithm to obtain two ciphertext vectors, v_{row} and v_{col} , where v_{row} and v_{col} represent ciphertext vectors with 1s at the r -th and c -th positions, respectively, and 0s elsewhere. The server calculates the matrix-vector product $M_c = M \times v_{col}$, involving plaintext-ciphertext multiplication and ciphertext-ciphertext addition. M_c represents a vector containing all elements from the c -th column of matrix M . The server then needs to perform similar steps for M_c and v_{row} . However, this process involves ciphertext-ciphertext multiplication, and since all elements of M_c are ciphertexts, it might be too large to encrypt into a single ciphertext. Therefore, M_c is split into F blocks, forming an $\sqrt{N} \times F$ matrix, which is then multiplied by v_{row} to obtain F ciphertexts. For these F ciphertexts, they need to be decrypted and combined into $\text{ENCRYPT}(M[r, c], pk)$, which is then decrypted to obtain the desired $M[r, c]$ for the client. Algorithms 10–12 illustrated the pseudocode for these processes where the hypercube dimension $d = 2$. In Algorithm 12, $M^1[r, c], \dots, M^F[r, c]$ represent the F components of $\text{ENCRYPT}(M[r, c], pk)$. We have omitted some details. For more information, please refer to [4].

Algorithm 10 QUERY

Input: $[r, c], pk$

Output: q_{row}, q_{col}

1: $q_{row} = \text{ENCRYPT}(x^r, pk), q_{col} = \text{ENCRYPT}(x^c, pk)$

Algorithm 11 EXPAND

Input: q_{row}, q_{col}, pk

Output: v_{row}, v_{col}

1: $v_{row} = [\text{ENCRYPT}(0, pk), \dots, \text{ENCRYPT}(0, pk), \text{ENCRYPT}(1, pk), \text{ENCRYPT}(0, pk), \dots, \text{ENCRYPT}(0, pk)], v_{row}[r] = \text{ENCRYPT}(1, pk)$
 2: $v_{col} = [\text{ENCRYPT}(0, pk), \dots, \text{ENCRYPT}(0, pk), \text{ENCRYPT}(1, pk), \text{ENCRYPT}(0, pk), \dots, \text{ENCRYPT}(0, pk)], v_{col}[c] = \text{ENCRYPT}(1, pk)$

Algorithm 12 MVPROD

Input: v_{row}, v_{col}, M, pk

Output: $[\text{ENCRYPT}(M^1[r, c], pk), \dots, \text{ENCRYPT}(M^F[r, c], pk)]$

1: $M_c = \text{HOMOMULT}(M, v_{col}, pk)$
 2: $[M_c^1, \dots, M_c^F] \leftarrow M_c$ // Split M_c into F blocks
 3: $[\text{ENCRYPT}(M^1[r, c], pk), \dots, \text{ENCRYPT}(M^F[r, c], pk)] = [\text{ENCRYPT}(M_c^1, pk), \dots, \text{ENCRYPT}(M_c^F, pk)] \times v_{row}$

4. Proposed Protocol

We propose a three-party secure and anonymous face verification scheme utilizing Facenet, the BFV encryption scheme, and SealPIR. The three parties include the client,

data server, and authentication server. Our scheme relies on the assumption that the authentication server is a trusted entity for the client and that there is no collusion between the servers. The scheme is divided into a registration phase and a verification phase.

4.1. Registration Phase

First, the authentication server generates the private key sk , public key pk , relinearization key rk , and Galois keys gk through encryption parameters. Then, it sends the public key pk , rk , and gk to the data server for encryption and computation, and sends pk to the clients for encryption. Please note that the public key sent to the clients can be the same. Therefore, in the following description, we will use one client as an example. The client then uses the MobileNetV1-based Facenet model to extract the facial feature vector v_A and subsequently sends v_A along with the corresponding index to the data server for registration. After receiving this information, the data server preprocesses the database into the matrix M . Note that, during the registration phase, the data server also generates a random number r^* (which can be generated using a pseudo-random function and a random seed) as a mask for the threshold t of Facenet, protecting the model information of the data server from being obtained by the authentication server. Please refer to Sections 4.2 and 4.3 for specific details. The pseudocode for the registration phase is illustrated in Algorithm 13. In Algorithm 13, C, DS, and AS represent the client, data server, and authentication server. The operations to be performed by each party are denoted after the colon “:”.

Algorithm 13 Registration phase

- 1: AS: $sk, pk, rk, gk \leftarrow \text{KENGEM}(\lambda, q, n)$
 - 2: AS: Send pk, rk, gk to DS and pk to C
 - 3: C: $v_A \leftarrow \text{Facenet}(\text{face image } A)$
 - 4: C: Send v_A and index to DS
 - 5: DS: Store (index, v_A) in the database
 - 6: DS: $M \leftarrow \text{Preprocess database}$
 - 7: DS: $r^* \leftarrow \text{Generate random number}$
 - 8: DS: $t^* = t^2 + r^*$
-

4.2. Verification Phase

First, similar to the registration phase, the client uses the Facenet model to extract the facial feature vector v_B from the face to be verified and encrypts it into the ciphertext ct_B . Simultaneously, the client obtains the ciphertexts q_{row} and q_{col} corresponding to the row and column indexes through the QUERY algorithm. Subsequently, the client sends q_{row} , q_{col} , and ct_B to the data server.

Upon receiving q_{row} and q_{col} , the data server uses the EXPAND algorithm to obtain the ciphertext vectors v_{row} and v_{col} . The data server then uses the MVPROD algorithm to obtain F ciphertexts sent to the authentication server. The authentication server decrypts and combines these F ciphertexts, obtaining $\text{ENCRYPT}(M[r, c], pk)$. $\text{ENCRYPT}(M[r, c], pk)$ represents the ciphertext of the facial feature vector corresponding to the face of interest for the client. Then, the authentication server sends $\text{ENCRYPT}(M[r, c], pk)$ to the data server for face similarity calculation. It is important to note that facial similarity, i.e., the distance $dist = \sqrt{\sum_{i=1}^n (M[r, c]_i - v_{Bi})^2}$ between two facial feature vectors, requires taking the square root. Calculating the square root poses a challenging problem for fully homomorphic encryption schemes, typically necessitating linear functions to approximate the square root, making the computation quite complex. Notably, comparing $dist$ with t is equivalent to comparing $dist^2$ and t^2 . By utilizing $dist^2$ and t^2 for comparison, we can avoid the complex process of square root computation. Therefore, the data server can perform HOMORSUB (denotes homomorphic subtraction, similar to HOMORADD) on $\text{ENCRYPT}(M[r, c], pk)$ and ct_B to obtain ct_s , then square ct_s , and perform $\log_2 n$ ROTATE and HOMORADD operations to obtain the ciphertext corresponding to $dist^2$, denoted as ct_{dist^2} .

Since the authentication server possesses the private key sk , the data server must send ct_{dist^2} and t^2 to the authentication server for comparison. However, this is not the desired scenario for the data server, as the authentication server is untrusted from the perspective of the data server. Allowing the authentication server to obtain $dist$ and t might leak certain information from the data server. As described in Section 4.1, the data server generates r^* as a mask for $dist$ and t during the registration phase. The data server computes $ct_{dist^*} = \text{HOMORADD}(ct_{dist^2}, r^*)$ and $t^* = t^2 + r^*$, then sends these values to the authentication server. These values are essentially treated as random values by the authentication server. Finally, the authentication server obtains the verification result and sends it to the client. The pseudocode for the verification phase is illustrated in Algorithm 14.

Algorithm 14 Verification phase

```

1: C:  $v_B \leftarrow \text{Facenet}(\text{face image } B)$ 
2: C:  $ct_B \leftarrow \text{ENCRYPT}(v_B, pk)$ 
3: C:  $q_{row}, q_{col} \leftarrow \text{QUERY}([r, c], pk)$ 
4: C: Send  $q_{row}, q_{col}, ct_B$  to DS
5: DS:  $v_{row}, v_{col} \leftarrow \text{EXPAND}(q_{row}, q_{col}, pk)$ 
6: DS:  $[\text{ENCRYPT}(M^1[r, c], pk), \dots, \text{ENCRYPT}(M^F[r, c], pk)] \leftarrow \text{MVPROD}(v_{row}, v_{col}, M, pk)$ 
7: DS: Send  $[\text{ENCRYPT}(M^1[r, c], pk), \dots, \text{ENCRYPT}(M^F[r, c], pk)]$  to AS
8: AS:  $M^1[r, c], \dots, M^F[r, c] \leftarrow \text{DECRYPT}(M^1[r, c], sk), \dots, \text{DECRYPT}(M^F[r, c], sk)$ 
9: AS:  $\text{ENCRYPT}(M[r, c], pk) \leftarrow \text{Combine } M^1[r, c], \dots, M^F[r, c]$ 
10: AS: Send  $\text{ENCRYPT}(M[r, c], pk)$  to DS
11: DS:  $ct_s \leftarrow \text{HOMORSUB}(\text{ENCRYPT}(M[r, c], pk), ct_B)$ 
12: DS:  $ct_{dist^2} \leftarrow \text{HOMORMULT}(ct_s, ct_s)$ 
13: DS:
14: for  $i = 1$  to  $\log_2 n$  do
15:    $ct_{dist^2} = \text{HOMORADD}(ct_{dist^2}, \text{ROTATE}(ct_{dist^2}, i, gk))$ 
16: end for
17: DS:  $ct_{dist^*} = \text{HOMORADD}(ct_{dist^2}, r^*)$ 
18: DS: Send  $ct_{dist^*}$  and  $t^*$  to AS
19: AS:  $dist^* = \text{DECRYPT}(ct_{dist^*}, sk)$ 
20: AS:
21: if  $dist^* \leq t^*$  then
22:    $result = 1$ 
23: else
24:    $result = 0$ 
25: end if
26: AS: Send  $result$  to C

```

4.3. Security Analysis

Table 2 illustrates the data possessed or obtained by each party during the verification phase. We will conduct the security analysis based on this table. Under the semi-honest security model, each party will faithfully adhere to the protocol but may attempt to obtain extra information.

For the client, concerning the matrix M obtained by preprocessing the data of the data server, the client needs to know the row and column indices corresponding to the data of interest. The pk possessed by the client can only be used for encrypting data. On the other hand, v_B , ct_B , q_{row} , and q_{col} are generated by the client and are not considered as additional information. The $result$ is the verification result the authentication server sends to the client. The client cannot obtain any other information from a single-face verification protocol, and can only determine whether the verification result is a success or failure. However, if the client's query attempts are not limited, malicious clients could potentially attempt to determine the location of the index corresponding to their facial information by querying all indices or ascertaining whether the facial information exists in the data

server's database. Therefore, the data server can prevent such attacks by restricting the number of query failures allowed per client.

For the data server, the pk can only be used for encrypting data, while rk and gk are exclusively utilized for ciphertext computations. M , r^* , and t^* are generated by the data server itself and are not considered as additional information. Furthermore, from the perspective of the data server, whether it receives v_{row} , v_{col} and $\text{ENCRYPT}(M[r, c], pk)$ from the client and authentication server, or obtains $[\text{ENCRYPT}(M^1[r, c], pk), \dots, \text{ENCRYPT}(M^F[r, c], pk)]$, ct_s , ct_{dist^2} , and ct_{dist^*} after ciphertexts computations, all these data are ciphertexts. Since the data server does not possess the private key sk , it cannot decrypt these data and thus cannot obtain additional information.

For the authentication server, sk , pk , rk , and gk are generated by executing the KEYGEN algorithm and are not considered additional information. Since the private key sk is only stored on the authentication server, the client does not need to bear the risk of private key leakage. The authentication server can obtain $\text{ENCRYPT}(M[r, c], pk)$ from $M^1[r, c]$, $\dots$, $M^F[r, c]$, where $\text{ENCRYPT}(M[r, c], pk)$ represents the ciphertext of facial feature data of a client. The authentication server only needs to decrypt this ciphertext to obtain the corresponding plaintext. However, since the client trusts the authentication server, this information does not compromise the client's security. Similarly, the $result$ possessed by the authentication server does not affect the client's security. The ct_{dist^*} and t^* obtained by the authentication server from the data server have been randomized. Apart from the verification result $result$, no additional information about the data server can be inferred from them.

In conclusion, for a single face verification process, the client and the authentication server send only ciphertext to the data server, ensuring that the data server cannot obtain information about the client's face image and cannot determine which entry the client wants to match, thus ensuring security and anonymity. For the risk posed by malicious clients conducting excessive queries, the data server can prevent such attacks by limiting the number of failed queries. Therefore, under the assumption that the client trusts the authentication server, our scheme ensures anonymity for the client and security for all parties involved.

Table 2. The data possessed or obtained by each party during the verification phase.

Party	C	DS	AS
Data	$pk, v_B, ct_B, q_{row}, q_{col}, result$	$pk, rk, gk, M, v_{row}, v_{col},$ $[\text{ENCRYPT}(M^1[r, c], pk), \dots,$ $\text{ENCRYPT}(M^F[r, c], pk)],$ $\text{ENCRYPT}(M[r, c], pk), ct_s,$ $ct_{dist^2}, r^*, ct_{dist^*}, t^*$	$sk, pk, rk, gk, M^1[r, c], \dots,$ $M^F[r, c], \text{ENCRYPT}(M[r, c],$ $pk), ct_{dist^*}, t^*, result$

5. Evaluation

Inter(R) Core(TM) i7-12700 CPU @3.60 GHz is used in our experiments. We use the Facenet library based on MobileNetV1 to extract facial feature vectors. During the training of the Facenet network, we utilize the CASIA-WebFace [45] dataset as the training set. The LFW [46] dataset serves as the test set and the database of the data server in our experiments. LFW comprises facial images of 5749 distinct individuals. We employ the BFV encryption scheme from the Seal fully homomorphic encryption library, as well as the SealPIR library based on the BFV encryption scheme. The security level of our scheme is 128 bits. The accuracy of face verification is solely determined by the Facenet model and the noise level of the ciphertext during encryption. It is unrelated to the security and anonymity discussed in this paper. Therefore, the comparison with related work in this scheme does not involve accuracy comparison.

5.1. Parameter Selection

Due to the limitation of the BFV encryption scheme, which only supports the encryption of integer-type data, the face feature vectors extracted by Facenet are real-valued vectors with 128 dimensions. To ensure that the BFV scheme can encrypt this data and minimize the impact on precision, we quantize this vector by multiplying it with an amplification vector and then rounding. In this experiment, the amplification factor is set to 100. We save the processed vectors as uint8 type, with each element size being 232 bytes. Therefore, the database in the data server consists of a total of 5749 data entries, each with a size of 232 bytes. The selection of various encryption parameters for the BFV encryption scheme, as well as the sizes of the public key, relinearization key, and Galois keys generated by the encryption parameters, are shown in Table 3. In Table 3, n , t , and q represent the polynomial degree, plaintext modulus, and ciphertext modulus sizes, respectively. pk , rk , and gk denote the sizes of the public key, relinearization key, and Galois keys, respectively.

Table 3. The sizes of various encryption parameters and keys generated by the BFV encryption scheme.

n	t (bit)	q (bit)	pk (KB)	rk (KB)	gk (KB)
4096	20	109 (36 + 36 + 37)	131	271	1887

5.2. Preprocess

Our scheme's registration phase can be preprocessed, reducing the computation and communication load during the online phase. In our experiment, 5749 clients trusting the same authentication server will register, uploading their face feature vectors and corresponding indices, which will be stored in the database of the data server. It is important to note that when the data server preprocesses the database, considering we have 5749 data entries and the polynomial degree n of 4096, using a single ciphertext cannot index the entire database. Therefore, we set $d = 2$, constructing the database as a 2-dimensional hypercube.

We use SealPIR to encode the database elements in the coefficients of BFV plaintext polynomials. Since the BFV plaintext modulus t is 20 bits, a coefficient of a BFV plaintext polynomial can accommodate 20 bits of data. Considering an element's size is 232 bytes, it requires $\lceil \frac{8 \times 232}{20} \rceil = 93$ coefficients. With a BFV polynomial degree $n = 4096$, a plaintext polynomial can represent at most $\lfloor \frac{4096}{93} \rfloor = 44$ elements. Each plaintext contains $44 \times 93 = 4092$ coefficients, and the size of each plaintext is $44 \times 232 = 10,208$ bytes. Therefore, the entire database requires $\lceil \frac{5749}{44} \rceil = 131$ BFV plaintexts for representation.

5.3. Online

Due to the fact that the registration phase of our scheme can be performed in advance, but the verification phase must be performed online, the main factor affecting the performance of this protocol is the verification process.

Firstly, the client needs to perform one face feature extraction operation, one ENCRYPT operation, and one QUERY operation. Afterward, a total of three ciphertexts need to be sent to the data server. The data server needs to perform one MVPROD operation to obtain F ciphertexts and send them to the authentication server. In our experiments, the size of F is 4. The authentication server decrypts and combines these four ciphertexts into one ciphertext, which is then sent to the data server. The data server then performs distance calculation and randomization on ciphertexts, obtains the result, and sends the result ct_{dist^*} and the threshold t^* to the authentication server. Finally, the authentication server decrypts and compares to obtain the verification result, which is then sent to the client. It should be noted that when the parties communicate, the sender needs to serialize the ciphertext into binary data before sending it to the receiver. Upon receiving this binary data, the receiver needs to deserialize it before performing subsequent operations on the ciphertext.

Table 4 illustrates the operations that each party needs to perform in the verification phase. In Table 4, “Facenet” represents the operation of extracting facial feature vectors in the Facenet model. “Serialize” and “Deserialize” indicate the operations of serializing and deserializing a ciphertext, respectively. Operations enclosed in parentheses are considered as a whole; for example, “(1 EXPAND + 1 MVPROD)” denotes the process where the data server obtains four ciphertexts using q_{row} , q_{col} , and M . “(4 DECRYPT + combine ciphertexts)” represents the authentication server obtaining $ENCRYPT(M[r, c], pk)$ from the four ciphertexts sent by the data server. “(1 HOMORSUB + 8 HOMORADD + 7 ROTATE)” indicates the data server’s process of obtaining ct_{dist}^* through homomorphic calculations. Table 5 elaborates on the time required for these operations and the total time for the verification phase. Table 6 explains the data types involved in the communication between parties. Table 7 outlines the sizes of the data in Table 6. Table 8 shows each party’s number of communications and total communication volume.

Table 4. The operations performed by each party during the verification phase.

Party	C	DS	AS
Operation	1 Facenet + 1 ENCRYPT + 1 QUERY + 3 Serialize	3 Deserialize + (1 EXPAND + 1 MVPROD) + (1 HOMORSUB + 8 HOMORADD + 7 ROTATE) + 5 Serialize	5 Deserialize + (4 DECRYPT + combine ciphertexts) + 1 Serialize + 1 DECRYPT

Table 5. The time required to perform each operation. The “Total” represents the overall time required for one face verification process.

Operation	Facenet	ENCRYPT	DECRYPT	QUERY	Serialize	Deserialize	1 EXPAND + 1 MVPROD	1 HOMORSUB + 8 HOMORADD + 7 ROTATE	4 DECRYPT + Combine Ciphertexts	Total
Time (ms)	6.24	2.21	0.29	3.32	2.99	0.36	32.76	8.67	1.63	84.91

Table 6. The type of data that is communicated between the various parties.

Communication Direction	C → DS	DS → AS	AS → DS	AS → C
Data type	q_{row}, q_{col}, ct_B	$ENCRYPT(M^1[r, c], pk), \dots, ENCRYPT(M^4[r, c], pk), ct_{dist}^*, t^*$	$ENCRYPT(M[r, c], pk)$	$result$

Table 7. The sizes of the data in Table 6.

Data Type	q_{row}, q_{col}	ct_B	$ENCRYPT(M^1[r, c], pk), \dots, ENCRYPT(M^4[r, c], pk)$	ct_{dist}^*	t^*	$ENCRYPT(M[r, c], pk)$	Result
Size (KB)	177	87	185	87	< 1	87	< 1

Table 8. The number of communications and total communication volume of each party.

Party	C	DS	AS
Number of communications	2	3	4
Total communications volume (KB)	264	359	359

5.4. Result

Due to the limited computational and communication capabilities of the client, our scheme primarily focuses on the client’s computational volume, the number of communications, and communication volume. In our scheme, the client’s computation time is 20.74 ms, requiring only one upload and one download operation, with a communication volume of 264 KB. Our scheme achieves a secure and anonymous face verification process in 84.91 ms. Table 9 compares our proposed scheme and other biometric verification

schemes. In Table 9, “NoF” represents the number of parties. In the “Technologies” column, E, H, and Hi, respectively, represent Euclidean distance, Hamming distance, and histogram intersection. “VD” represents the dimensionality of the feature vectors. “ n/m ” represents the polynomial degree or cyclotomic polynomial degree. “ q ” and “ k ” denote the ciphertext modulus and security level. “NoCC” and “CCV” represent the client’s number of communication rounds and communication volume. “Anonymity” indicates whether the scheme achieves anonymity. “N” and “Y” represent, respectively, the implementation and non-implementation of anonymity. “Time” represents the time required for one verification process. “\” indicates that the parameter is not present in the respective scheme or not explicitly mentioned.

Table 9. Comparison of our proposed scheme with other biometric verification schemes.

Scheme	NoP	Technologies	VD	n/m	q (bit)	λ	NoCC	CCV	Anonymity	Time
[5]	2	E + GH11	4000	$n = 2048$	22	70	2	393 MB	N	121.4 s
[6]	2	E + QHE	100	\	\	80	2	\	N	15.88 s
[47]	2	H + BGV	2400	$m = 8191$	40	80	2	\	N	0.37 s
[7]	3	H + Paillier	\	\	\	\	2	\	N	\
[8]	3	H + Paillier	256	\	\	80	2	\	N	0.71 s
[9]	2	E + BFV	1024	$n = 4096$	110	128	2	66 KB	N	11.42 ms
[10]	3	Hi + Paillier + GC	944	\	\	\	2	3 KB	N	\
[13]	2	E + QHE	128	\	\	80	n	33.64 KB	N	1.07 s
[11]	3	E + BFV / CKKS / NTRU	512	\	\	128	2	516 KB / 132 KB / 5.5 KB	N	0.72 s / 3.6 s / 50 ms
[14]	3	E + BGV + GC	512	$m = 6353$	\	126	2	118 KB	N	0.53 s
[15]	2	H + BFV	128	$m = 2048$	76	80	4	\	N	\
Ours	3	E + BFV + SealPIR	128	$n = 4096$	109	128	2	264 KB	Y	84.91 ms

Compared to the works of Troncoso-Pastoriza et al. [5], Im et al. [6], Cheon et al. [47], Ma et al. [8], and Huang and Wang [14], our scheme achieves a higher level of security with reduced verification time. Limited comparable information is provided in the works of Abidin [7] and Lin et al. [10]. In the scheme proposed by Boddeti [9], there is a potential for the client to acquire information that could pose a threat to the server. Compared to the BFV-based scheme proposed by Kolberg et al. [11], our scheme exhibits shorter processing times but incurs higher communication volume for the client. This is mainly due to the additional ciphertexts sent by the client for PIR. In the schemes proposed by Im et al. [13] and Sun et al. [15], the client has excessive communication. Compared to other works, one of the critical contributions of our proposed scheme is its ability to ensure anonymity during the verification process. Additionally, our scheme guarantees the security of information between the client and the data server, as well as between the data server and the authentication server. The absence of the private key at the client avoids the risk of private key exposure. Our scheme involves a relatively small number of communications, relatively low communication volumes, and little computation for the client. Furthermore, our approach can complete a verification process in a relatively short time.

6. Conclusions

This paper proposes a secure and anonymous face verification scheme using fully homomorphic encryption technology and SealPIR. The scheme not only protects the facial data of the client during the verification process from being accessed by the data server but also ensures that the data server cannot determine which facial data from the database the client intends to verify. Security analysis indicates that our scheme, under the assumption of client trust in the authentication server, guarantees the security of all parties involved and the anonymity of the verification process. In our scheme, the client’s computation time is only 20.74 ms, and it involves only two communications with a communication volume of 264 KB. Experimental results demonstrate that our scheme can accomplish a secure and anonymous face verification process in 84.91 ms. Future work may involve extending our scheme to batch multiple face verification requests for the same client.

Author Contributions: Conceptualization, X.W.; methodology, X.W.; software, X.W.; validation, X.W.; formal analysis, X.W.; investigation, X.W.; resources, X.W.; data curation, X.W.; writing—original draft preparation, X.W.; writing—review and editing, X.W.; visualization, X.W.; supervision, P.L.; project administration, X.W.; funding acquisition, P.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Natural Science Foundation of Hebei Province (Grant number: F2019502173); the National Natural Science Foundation of China (Grant number: 61602173).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The CASIA-WebFace dataset and the LFW dataset used in this work are sourced from the following organizations, <https://paperswithcode.com/dataset/casia-webface> and <https://vis-www.cs.umass.edu/lfw/> (accessed on 23 September 2023).

Conflicts of Interest: The authors declare no competing interests.

Abbreviations

The following abbreviations are used in this manuscript:

NTRU	Number theory research unit
PIR	Private information retrieval
IT-PIR	Information-theoretic private information retrieval
CPIR	Computational private information retrieval
CNN	Convolutional neural network
BGV	A fully homomorphic encryption scheme proposed by Brakerski, Gentry, and Vaikuntanathan
BFV	A fully homomorphic encryption scheme proposed by Brakerski, Fan, and Vercauteren
CKKS	A fully homomorphic encryption scheme proposed by Cheon, Kim, Kim, and Song
TFHE	A fully homomorphic encryption over the torus
LWE	Learning with errors
RLWE	Ring learning with errors
SIMD	Single instruction multiple data
CRT	Chinese remainder theorem
PBC	Probabilistic batch codes

References

1. Wang, M.; Deng, W. Deep face recognition: A survey. *Neurocomputing* **2021**, *429*, 215–244. [\[CrossRef\]](#)
2. Marcolla, C.; Sucasas, V.; Manzano, M.; Bassoli, R.; Fitzek, F.H.; Aaraj, N. Survey on fully homomorphic encryption, theory, and applications. *Proc. IEEE* **2022**, *110*, 1572–1609. [\[CrossRef\]](#)
3. Chor, B.; Kushilevitz, E.; Goldreich, O.; Sudan, M. Private information retrieval. *J. ACM* **1998**, *45*, 965–981. [\[CrossRef\]](#)
4. Angel, S.; Chen, H.; Laine, K.; Setty, S. PIR with compressed queries and amortized query processing. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), IEEE, San Francisco, CA, USA, 20–24 May 2018; pp. 962–979.
5. Troncoso-Pastoriza, J.R.; González-Jiménez, D.; Pérez-González, F. Fully private noninteractive face verification. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 1101–1114. [\[CrossRef\]](#)
6. Im, J.H.; Choi, J.; Nyang, D.; Lee, M.K. Privacy-preserving palm print authentication using homomorphic encryption. In Proceedings of the 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), IEEE, Auckland, New Zealand, 8–12 August 2016; pp. 878–881.
7. Abidin, A. On privacy-preserving biometric authentication. In Proceedings of the Information Security and Cryptology: 12th International Conference, Inscrypt 2016, Beijing, China, 4–6 November 2016; Revised Selected Papers 12; Springer: Berlin/Heidelberg, Germany, 2017; pp. 169–186.
8. Ma, Y.; Wu, L.; Gu, X.; He, J.; Yang, Z. A secure face-verification scheme based on homomorphic encryption and deep neural networks. *IEEE Access* **2017**, *5*, 16532–16538. [\[CrossRef\]](#)
9. Boddeti, V.N. Secure face matching using fully homomorphic encryption. In Proceedings of the 2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS). IEEE, Darmstadt, Germany, 22–25 October 2018; pp. 1–10.
10. Lin, D.; Hilbert, N.; Storer, C.; Jiang, W.; Fan, J. UFace: Your universal password that no one can see. *Comput. Secur.* **2018**, *77*, 627–641. [\[CrossRef\]](#)

11. Kolberg, J.; Drozdowski, P.; Gomez-Barrero, M.; Rathgeb, C.; Busch, C. Efficiency analysis of post-quantum-secure face template protection schemes based on homomorphic encryption. In Proceedings of the 2020 International Conference of the Biometrics Special Interest Group (BIOSIG), IEEE, San Francisco, CA, USA, 16–18 September 2020; pp. 1–4.
12. Hoffstein, J.; Pipher, J.; Silverman, J.H. NTRU: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 267–288.
13. Im, J.H.; Jeon, S.Y.; Lee, M.K. Practical privacy-preserving face authentication for smartphones secure against malicious clients. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 2386–2401. [\[CrossRef\]](#)
14. Huang, H.; Wang, L. Efficient privacy-preserving face verification scheme. *J. Inf. Secur. Appl.* **2021**, *63*, 103055. [\[CrossRef\]](#)
15. Sun, D.; Huang, H.; Zheng, D.; Hu, H.; Bi, C.; Wang, R. Face security authentication system based on deep learning and homomorphic encryption. *Secur. Commun. Netw.* **2022**, *2022*, 7752292. [\[CrossRef\]](#)
16. Beimel, A.; Ishai, Y.; Kushilevitz, E.; Raymond, J.F. Breaking the $O(n/\log(1/(2k-1)))$ barrier for information-theoretic private information retrieval. In Proceedings of the The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002, Proceedings, IEEE, Vancouver, BC, Canada, 19 November 2002; pp. 261–270.
17. Demmler, D.; Herzberg, A.; Schneider, T. RAID-PIR: Practical multi-server PIR. In Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security, Scottsdale, AZ, USA, 7 November 2014; pp. 45–56.
18. Devet, C.; Goldberg, I.; Heninger, N. Optimally robust private information retrieval. In Proceedings of the 21st USENIX Security Symposium (USENIX Security 12), Bellevue, WA, USA, 6 August 2012; pp. 269–283.
19. Dvir, Z.; Gopi, S. 2-server PIR with subpolynomial communication. *J. ACM* **2016**, *63*, 1–15. [\[CrossRef\]](#)
20. Gentry, C.; Halevi, S.; Magri, B.; Nielsen, J.B.; Yakubov, S. Random-index PIR and applications. In Proceedings of the Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, 8–11 November 2021; Proceedings, Part III 19; Springer: Berlin/Heidelberg, Germany, 2021; pp. 32–61.
21. Goldberg, I. Improving the robustness of private information retrieval. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07), IEEE, Berkeley, CA, USA, 20–23 May 2007; pp. 131–148.
22. Shi, E.; Aqeel, W.; Chandrasekaran, B.; Maggs, B. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In Proceedings of the Advances in Cryptology—CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, 16–20 August 2021; Proceedings, Part IV 41; Springer: Berlin/Heidelberg, Germany, 2021; pp. 641–669.
23. Song, S.; Hayashi, M. Capacity of quantum private information retrieval with multiple servers. *IEEE Trans. Inf. Theory* **2020**, *67*, 452–463. [\[CrossRef\]](#)
24. Yeo, K. Lower bounds for (batch) pir with private preprocessing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 518–550.
25. Kushilevitz, E.; Ostrovsky, R. Replication is not needed: Single database, computationally-private information retrieval. In Proceedings of the 38th Annual Symposium on Foundations of Computer Science, IEEE, Miami Beach, FL, USA, 20–22 October 1997; pp. 364–373.
26. Cachin, C.; Micali, S.; Stadler, M. Computationally private information retrieval with polylogarithmic communication. In Proceedings of the Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; Proceedings 18; Springer: Berlin/Heidelberg, Germany, 1999; pp. 402–414.
27. Chang, Y.C. Single database private information retrieval with logarithmic communication. In Proceedings of the Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, 13–15 July 2004; Proceedings 9; Springer: Berlin/Heidelberg, Germany, 2004; pp. 50–61.
28. Gentry, C.; Ramzan, Z. Single-database private information retrieval with constant communication rate. In *International Colloquium on Automata, Languages, and Programming*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 803–815.
29. Sion, R.; Carbunar, B. On the computational practicality of private information retrieval. In *Network and Distributed Systems Security Symposium*; Internet Society: Geneva, Switzerland, 2007; p. 2006.
30. Melchor, C.A.; Barrier, J.; Fousse, L.; Killijian, M.O. XPIR: Private information retrieval for everyone. In *Proceedings on Privacy Enhancing Technologies*; HAL: Bangalore, India, 2016; pp. 155–174.
31. Mughees, M.H.; Chen, H.; Ren, L. OnionPIR: Response efficient single-server PIR. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, 15–19 November 2021; pp. 2292–2306.
32. Ahmad, I.; Yang, Y.; Agrawal, D.; El Abbadi, A.; Gupta, T. Addra: Metadata-private voice communication over fully untrusted infrastructure. In Proceedings of the 15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21), Virtual Event, 14–16 July 2021.
33. Mughees, M.H.; Ren, L. Vectorized batch private information retrieval. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), IEEE, San Francisco, CA, USA, 21–25 May 2023; pp. 437–452.
34. Schroff, F.; Kalenichenko, D.; Philbin, J. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 815–823.
35. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.

36. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the forty-first annual ACM symposium on Theory of computing, New York, NY, USA, 31 May–2 June 2009; pp. 169–178.
37. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory (TOCT)* **2014**, *6*, 1–36. [[CrossRef](#)]
38. Fan, J.; Vercauteren, F. *Somewhat Practical Fully Homomorphic Encryption*; Cryptology ePrint Archive: Bellevue, WA, USA, 2012.
39. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In Proceedings of the Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; Proceedings, Part I 23; Springer: Berlin/Heidelberg, Germany, 2017; pp. 409–437.
40. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. TFHE: Fast fully homomorphic encryption over the torus. *J. Cryptol.* **2020**, *33*, 34–91. [[CrossRef](#)]
41. *Microsoft SEAL (Release 4.0)*; Microsoft Research: Redmond, WA, USA, 2022. Available online: <https://github.com/Microsoft/SEAL> (accessed on 13 October 2023).
42. Smart, N.P.; Vercauteren, F. Fully homomorphic SIMD operations. *Des. Codes Cryptogr.* **2014**, *71*, 57–81. [[CrossRef](#)]
43. Brakerski, Z.; Gentry, C.; Halevi, S. Packed ciphertexts in LWE-based homomorphic encryption. In Proceedings of the Public-Key Cryptography–PKC 2013: 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, 26 February–1 March 2013; Proceedings 16; Springer: Berlin/Heidelberg, Germany, 2013; pp. 1–13.
44. Gentry, C.; Halevi, S.; Smart, N.P. Fully homomorphic encryption with polylog overhead. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 465–482.
45. Yi, D.; Lei, Z.; Liao, S.; Li, S.Z. Learning face representation from scratch. *arXiv* **2014**, arXiv:1411.7923.
46. Huang, G.B.; Mattar, M.; Berg, T.; Learned-Miller, E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In Proceedings of the Workshop on Faces in ‘Real-Life’ Images: Detection, Alignment, and Recognition, Technical Report 07-49, Amherst, MA, USA, 16 September 2008.
47. Cheon, J.H.; Chung, H.; Kim, M.; Lee, K.W. *Ghostshell: Secure Biometric Authentication Using Integrity-Based Homomorphic Evaluations*; Cryptology ePrint Archive: Lyon, France, 2016.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.