Pura: An Efficient Privacy-Preserving Solution for Face Recognition

Guotao Xu, Bowen Zhao*, *Member, IEEE*, Yang Xiao*, *Member, IEEE*, Yantao Zhong, Liang Zhai, Qingqi Pei, *Senior Member, IEEE*

***Abstract*—Face recognition is an effective technology for identifying a target person by facial images. However, sensitive facial images raises privacy concerns. Although privacy-preserving face recognition is one of potential solutions, this solution neither fully addresses the privacy concerns nor is efficient enough. To this end, we propose an efficient privacy-preserving solution for face recognition, named Pura, which sufficiently protects facial privacy and supports face recognition over encrypted data efficiently. Specifically, we propose a privacy-preserving and non-interactive architecture for face recognition through the threshold Paillier cryptosystem. Additionally, we carefully design a suite of underlying secure computing protocols to enable efficient operations of face recognition over encrypted data directly. Furthermore, we introduce a parallel computing mechanism to enhance the performance of the proposed secure computing protocols. Privacy analysis demonstrates that Pura fully safeguards personal facial privacy. Experimental evaluations demonstrate that Pura achieves recognition speeds up to 16 times faster than the state-of-the-art.**

***Index Terms*—Face recognition, secure computing, homomorphic encryption, threshold Paillier cryptosystem, privacy protection.**

I. Introduction

FACE recognition is a practical biometric technology with widespread applications across various fields, including attendance access control [1], security [2], and finance [3]. Technically, face recognition identifies and verifies individuals based on their unique facial features [4]. These features are captured by cameras and transformed into feature vectors through a face recognition model [5]. The recognition result is then determined by measuring the similarity (e.g., Euclidean distance, cosine similarity) between a feature vector and stored database entries [6].

Outsourcing face recognition tasks is a common way to enhance flexibility [7], however, it raises significant privacy concerns. To outsource a face recognition task, the client device (e.g., smartphone, camera) captures personal facial images and sends the images or feature vectors to a cloud server, where the cloud server executes face recognition computations and returns the recognition result [8], [9]. Outsourcing face recognition tasks reduces the computational burden on the client device by utilizing the high-performance computing capabilities of the cloud server. Nevertheless, this practice introduces considerable privacy risks. The transmission and storage of sensitive data (i.e., facial images, feature vectors) by the cloud server may lead to data leakage [10]. A notable example is the iCloud celebrity photo leak incident, which results in the unauthorized release of personal images [11]. In various privacy-sensitive scenarios, leaky images or feature vectors threaten personal information security [12].

Privacy-preserving face recognition (PPFR) is a potential solution that balances the strengths and concerns of outsourcing recognition tasks. A PPFR scheme enables the cloud server to perform face recognition without leaking facial data including facial images and feature vectors. To safeguard privacy, PPFR usually protects facial data through homomorphic encryption [13], secret sharing [14], or differential privacy [15]. In this case, the cloud server is enforced to perform face recognition operations on encrypted or confused data [16], [17]. However, current PPFR methods still face several limitations.

Existing PPFR solutions cannot always fully safeguard personal facial data and require interactions between the client and the cloud server. Current work, such as [18]–[20] requires client-server interactions during the recognition computation process, introducing computational and communication burdens on the client side. Furthermore, current PPFR solutions leak facial data like feature vectors to the cloud server. Most PPFR solutions adopt homomorphic encryption (HE) to support direct computations over encrypted data. Unfortunately, some HE-based PPFR solutions [21]–[23] can not fully safeguard facial data because the cloud server obtains the plaintext results of the intermediate calculations, which means the cloud server can infer more information, such as feature vectors.

Moreover, existing PPFR approaches struggle to support effective recognition computations while maintaining recognition accuracy. On the one hand, PPFR solutions [20], [23] introduce new approaches for computing the similarity, such as the square of Euclidean distance and cosine similarity. However, they fall short in effectively performing the comparison operation, as they rely on plaintext for comparison. On the other hand, maintaining the same accuracy as schemes without privacy protection remains a difficulty. For instance,

G. Xu and B. Zhao are with the Guangzhou Institute of Technology, and Shaanxi Key Laboratory of Blockchain and Secure Computing, Xidian University, Guangzhou, China. E-mail: guotaoxu@stu.xidian.edu.cn, bwinzhao@gmail.com

Yang Xiao is with the School of Cyber Engineering, and the Engineering Research Center of Trusted Digital Economy, Universities of Shaanxi Province, Xidian University, Xi'an, China. E-mail: yxiao@xidian.edu.cn

Yantao Zhong, China Resources Intelligent Computing Technology Co., Ltd., Guangzhou, China. E-mail: zhongyantao@126.com

Liang Zhai, Chinese Academy of Surveying and Mapping, Beijing, China. E-mail: zhailiang@casm.ac.cn

Qingqi Pei is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China. E-mail: qqpei@mail.xidian.edu.cn

Corresponding author: Bowen Zhao, Yang Xiao

the solution [24] employs approximation during computation, which results in a loss of accuracy. Additionally, some existing PPFR solutions [25], [26] based on differential privacy enhance privacy protection by introducing noise, but this also compromises the usability of facial data and reduces recognition accuracy.

In general, existing PPFR solutions are limited by low efficiency due to complex computations and the inefficient use of computing resources. Although homomorphic encryption allows for computations over ciphertexts, recognition typically involves processing large amounts of data, which increases the computational overhead. To alleviate this dilemma, solutions [21], [23] propose batch computations over ciphertexts. However, they still suffer from high computation costs, as operations like multiplication and minimum are still expensive and time-consuming. Furthermore, the distribution of server computing power is often suboptimal. In some multi-server PPFR solutions [27], [28], one server always needs to wait for the calculation results from another one, which results in a waste of computational resources.

To avoid the above limitations, in this work, we innovatively propose an efficient privacy-preserving solution for face recognition, named Pura[1]. Roughly speaking, we propose a privacy-preserving face recognition framework that is non-interactive and ensures no privacy leakage. To enable effective computations over ciphertexts without degrading accuracy, we design a suite of secure computing protocols. Additionally, by introducing a parallel computing mechanism, we improve the efficiency of the proposed Pura. The contributions of this paper are summarized in three folds:

- We propose a novel framework for non-interactive and privacy-preserving face recognition falling in a twin-server architecture. The proposed framework enables cloud servers to perform privacy-preserving face recognition operations without frequent client-server interactions.
- We carefully design a suite of secure computing protocols that support effective recognition operations without sacrificing accuracy. The proposed protocols, including a batch secure square protocol (`BatchSquare`) and secure minimum protocols (2-`SMIN` and n-`SMIN`), enable valuable operations over ciphertexts, and support Pura to effectively perform the privacy-preserving face recognition task.
- We introduce an efficient parallel computing mechanism to enhance the computational efficiency of the servers. Specifically, both data storage and encrypted data computation are assigned to two servers simultaneously. Experimental evaluations demonstrate the efficiency of the proposed secure computing protocols and Pura.

The rest of this work is organized as follows. Section II briefly reviews the related work on privacy-preserving face recognition, and Section III then introduces preliminaries. After that, we define the system model and threat model in Section IV. We carefully design a suite of secure computing protocols in Section V. In Section VI, we elaborate on the design of Pura, including problem formulation and the workflow of Pura. Rigorous privacy analysis is given in Section VII. In Section VIII, the experimental results and a comprehensive analysis are shown. Finally, we summarize this work in Section IX.

[1]Pura: an efficient privacy-preserving solution for face recognition

II. Related Work

In this section, we briefly review the related work on privacy-preserving face recognition.

Privacy-preserving face recognition has been extensively researched and has shown rapid development in recent years. Partially homomorphic encryption (PHE) allows specific computations over ciphertexts. Erkin *et al.* [18] first integrated PHE into face recognition and proposed a privacy-preserving face recognition system. Subsequently, Sadeghi *et al.* [19] combined PHE with garbled circuit (GC) to enhance the performance of the work [18]. Huang *et al.* [21] further optimized previous work of PPFR based on PHE and GC. However, the above solutions require client-server interactions during the recognition phase. Chun *et al.* [27] introduced a more secure and outsourceable privacy-preserving biometric authentication protocol. Nevertheless, this approach raises privacy concerns as the server gains knowledge of the recognition result, potentially revealing whether the input face matches any person in the database.

Since fully homomorphic encryption (FHE) supports more mathematical operations over encrypted data than PHE, several solutions based on FHE have been proposed lately. Xiang *et al.* [22] firstly outsourced computation and presented a privacy-preserving face recognition protocol based on FHE. Although [22] encrypts the input facial data during face recognition, it stores the face database in plaintext on the server. Drozdowski *et al.* [23] presented a PPFR protocol for a three-party PPFR architecture using FHE. However, the private key is held by one of the servers, thus the server is able to decrypt the ciphertexts during computation. Huang *et al.* [28] proposed the state-of-the-art PPFR protocol based on FHE and GC. Unfortunately, the work [28] utilizes the column-wise strategy when storing the face database, resulting in a time-consuming process for dynamic updates of the database.

To the best of our knowledge, there are privacy-preserving biometric recognition solutions also using secret sharing [24] and differential privacy [25] to protect privacy. However, in the work [24], the server obtains the input access pattern, which can retrieve a record from the database a record that matches the input face data. In the solution [25], recognition accuracy is degraded due to the noise introduced.

To summarize, existing PPFR solutions still suffer from privacy leakage and poor efficiency. Our work, Pura, addresses the aforementioned issues. Pura is non-interactive and efficient. Besides, Pura leaks nothing about facial data and maintains the same precision as schemes without privacy protection. Additionally, Pura supports the update of the face database dynamically and allows for batch computations in parallel.

III. Preliminaries

In this section, we introduce the threshold Paillier cryptosystem [29] and the traditional face recognition approach

[30]. To improve clarity, we denote $[\![m]\!]$ as the ciphertext of a plaintext m. In addition, "$\rightarrow$" means an output operation. And $\overset{\$}{\leftarrow}$ represents a random selection operation.

A. Threshold Paillier Cryptosystem

In this work, we consider the (2,2)-threshold Paillier cryptosystem [31], which consists of the following polynomial time algorithms:

N Generation (NGen): NGen takes a security parameter κ as input and outputs (N, P, Q, p, q). Formally, NGen is comprised of the following steps:

1) Randomly generate $\frac{l(\kappa)}{2}$-bit odd primes p and q, where $l(\kappa)$ represents the bit length of the private key of the Paillier cryptosystem and $l(\kappa) = 4\kappa$;
2) Randomly generate $(\frac{n(\kappa)-l(\kappa)}{2} - 1)$-bit odd integers p' and q', where $n(\kappa)$ refers to the bit length of the modulus N for Paillier;
3) Calculate $P = 2pp' + 1$ and $Q = 2qq' + 1$;
4) If p, q, p', q' are not co-prime, or P or Q is not a prime, then re-generate (p, q, P, Q) from step 1);
5) Calculate $N = PQ$, then output (N, P, Q, p, q).

Key Generation (KeyGen): KeyGen takes a security parameter κ as input and generates a public key pk and a private key sk. KeyGen calls NGen to get (N, P, Q, p, q) first. Then, KeyGen sequentially calculates $\alpha = pq$, $\beta = \frac{(P-1)(Q-1)}{4pq}$, and $h = -y^{2\beta} \mod N$, where $y \overset{\$}{\leftarrow} \mathbb{Z}_N^*$. The public key is denoted by $pk = (h, N)$ and the private key is denoted by $sk = \alpha$. Particularly, the private key sk is split into two threshold keys, denoted by sk_1 and sk_2, such that $sk_1 + sk_2 = 0 \mod 2\alpha$ and $sk_1 + sk_2 = 1 \mod N$. sk_1 is set as a random number with σ bits and $sk_2 = ((2\alpha)^{-1} \mod N) \cdot (2\alpha) - sk_1 + \eta \cdot 2\alpha \cdot N$, where $\eta \in \mathbb{Z}$.

Encryption (Enc): Enc inputs a plaintext $m \in \mathbb{Z}_N$ and pk, and outputs a ciphertext $[\![m]\!] \in \mathbb{Z}_{N^2}^*$, denoted as

$$\begin{aligned} [\![m]\!] &\leftarrow \texttt{Enc}(pk, m) \\ &= (1+N)^m \cdot (h^r \mod N)^N \mod N^2, \end{aligned} \tag{1}$$

where $r \overset{\$}{\leftarrow} \{0,1\}^{l(\kappa)}$.

Decryption (Dec): Dec inputs a ciphertext $[\![m]\!] \in \mathbb{Z}_{N^2}^*$ and sk, and outputs a plaintext $m \in \mathbb{Z}_N$, formulated as

$$\begin{aligned} m &\leftarrow \texttt{Dec}(sk, [\![m]\!]) \\ &= \left(\frac{([\![m]\!]^{2\alpha} \mod N^2) - 1}{N} \mod N\right) \cdot (2\alpha)^{-1} \mod N. \end{aligned} \tag{2}$$

Partial Decryption (PDec): PDec inputs a ciphertext $[\![m]\!] \in \mathbb{Z}_{N^2}^*$ and a partially private key sk_i ($i \in \{1, 2\}$), and outputs a partially decrypted ciphertext $[\![M_i]\!] \in \mathbb{Z}_{N^2}^*$, defined as

$$[\![M_i]\!] \leftarrow \texttt{PDec}(sk_i, [\![m]\!]) = [\![m]\!]^{sk_i} \mod N^2. \tag{3}$$

Threshold Decryption (TDec): TDec inputs a pair of partially decrypted ciphertexts $[\![M_1]\!], [\![M_2]\!] \in \mathbb{Z}_{N^2}^*$, and outputs a plaintext $m \in \mathbb{Z}_N$, described as

$$\begin{aligned} m &\leftarrow \texttt{TDec}([\![M_1]\!], [\![M_2]\!]) \\ &= \frac{([\![M_1]\!] \cdot [\![M_2]\!] \mod N^2) - 1}{N} \mod N. \end{aligned} \tag{4}$$

The (2,2)-threshold Paillier cryptosystem supports additive homomorphism and scalar-multiplication homomorphism.

- **Additive homomorphism**: $\texttt{Dec}(sk, [\![m_1]\!] \cdot [\![m_2]\!]) = \texttt{Dec}(sk, [\![m_1 + m_2 \mod N]\!])$.
- **Scalar-multiplication homomorphism**: $\texttt{Dec}(sk, [\![m]\!]^u) = \texttt{Dec}(sk, [\![u \cdot m \mod N]\!]), u \in \mathbb{Z}_N$.

B. Face Recognition

Face recognition checks whether a probe face matches any person in a face database of multiple persons or not [32]. In practice, the facial image is usually transformed into an n-dimensional feature vector extracted by a deep learning model, e.g., Facenet [33]. Face recognition usually includes the registration phase and the recognition phase [34], [35].

The registration phase involves storing multiple feature vectors, also known as face templates, on the server. Formally, an n-dimensional feature vector database of Υ persons is formulated as a matrix with Υ rows and $n+1$ columns:

$$\mathcal{D}_{\Upsilon \times (n+1)} = \begin{pmatrix} ID_1 & v_{1,1} & v_{1,2} & \cdots & v_{1,n} \\ ID_2 & v_{2,1} & v_{2,2} & \cdots & v_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ ID_\Upsilon & v_{\Upsilon,1} & v_{\Upsilon,2} & \cdots & v_{\Upsilon,n} \end{pmatrix},$$

where ID means the identification of a person.

The recognition phase consists of calculating the distances between the probe feature vector and the feature vector database, and comparing the minimum distance to a threshold value. A probe feature vector is denoted as $p = (p_1, p_2, \cdots, p_n)$. In this work, we utilize the square of Euclidean distance [36] as the distance metric. The square of Euclidean distances between the probe feature vector p and all feature vectors in database $\mathcal{D}$ are defined as:

$$d_i^2 = \sum_{j=1}^{n} (p_j - v_{i,j})^2, 1 \le i \le \Upsilon. \tag{5}$$

After that, the validity of the recognition result is determined by searching for the minimum distance d_{min} among $\{d_1^2, d_2^2, \cdots, d_\Upsilon^2\}$ and then comparing d_{min} with a threshold. If d_{min} is not greater than the threshold, the recognition is successful, otherwise, the recognition is regarded as failed.

IV. System Model and Threat Model

In this section, we illustrate our system model and threat model of Pura.

A. System Model

As depicted in Fig. 1, our proposed Pura consists of an organization, multiple users, and two servers S_1 and S_2.

- **Organization.** The organization owns a database of feature vectors. The organization calls KeyGen to generate a public key pk, a private key sk, and threshold keys (sk_1, sk_2). The organization encrypts the whole feature vector database under pk and horizontally splits the encrypted database into two parts, denoted as $[\![\mathcal{D}]\!]_1$ and $[\![\mathcal{D}]\!]_2$. The organization distributes pk, $(pk, sk_1, [\![\mathcal{D}]\!]_1)$, and $(pk, sk_2, [\![\mathcal{D}]\!]_2)$ to the users, S_1, and S_2, respectively.

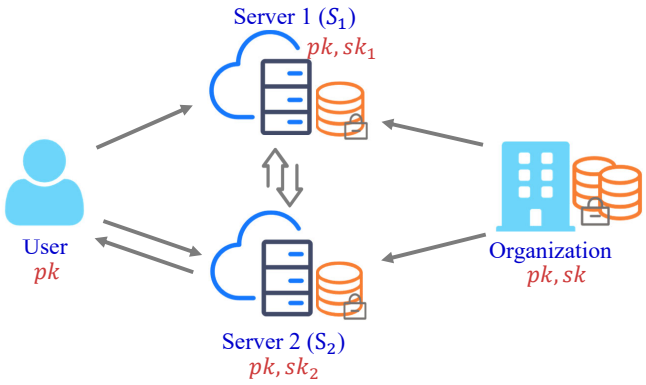


Fig. 1: System model.

- **User**. The user is capable of capturing personal facial images. The user extracts a facial image into a feature vector p. To preserve privacy, p is encrypted under pk and is sent to both S_1 and S_2. Finally, the user obtains information from S_2 and recovers the recognition result.
- **Twin servers.** S_1 and S_2 both store a portion of the encrypted feature vector database. They are responsible for a part of the recognition computations, respectively. Meanwhile, they also provide online computation services for each other. When a user requests a recognition service, S_1 and S_2 jointly and parallelly perform face recognition operations over ciphertexts directly.

B. Threat Model

In this work, there is one type of adversary including the twin servers S_1 and S_2. Following previous work [31], [37]–[40], we suppose that S_1 and S_2 are semi-honest (or say honest-but-curious) and non-colluding. This means S_1 and S_2 follow the protocols strictly but they attempt to infer a user's private information, such as the face image and the recognition result. Besides, we consider the organization to be fully trusted and does not collude with other entities. We assume that the user is honest.

We consider the twin servers S_1 and S_2 to be provided by different or even competitive cloud service providers. Due to the commercial competition between service providers, S_1 and S_2 have no incentive to collude, as doing so would risk their own interests. The collusion between S_1 and S_2 implies that one server can access the private data of the other server, such as the partially private key, which will leave evidence for the competitor and even cause business losses. Therefore, collusion between the twin servers provided by different cloud service providers is unrealistic.

V. Secure Computing Protocols

To support face recognition operations that do not disclose privacy, we carefully design a suite of secure computing protocols including batch secure multiplication (`BatchSMUL`), batch secure square (`BatchSquare`), secure 2-minimum (2-`SMIN`), and secure n-minimum (n-`SMIN`).

A. `BatchSquare`

FREED [41] has constructed a `BatchSMUL` protocol based on the threshold Paillier cryptosystem. However, the work [41] only supports batch secure multiplication over non-negative numbers. To extend batch secure multiplication to support negative numbers, we proposed a novel batch secure batch multiplication (`BatchSMUL`) protocol.

We introduce a shared constant δ to convert the negative numbers into non-negative numbers. Note that, δ is set as 0 if all the plaintext values are non-negative, or else δ is set as the absolute value of the smallest number. `BatchSMUL` can perform secure multiplication for up to $\lfloor |N|/2|L| \rfloor$ pairs of ciphertexts at the same time [41], where L is also a shared constant and $L \geq 2^{\sigma+2}$, and σ is a secure parameter and $2^{\sigma} \gg 2^{\ell}$, where ℓ is a constant that controls the domain size of plaintext. `BatchSMUL` takes two groups of ciphertexts $[\![x_1]\!], \cdots, [\![x_s]\!]$ and $[\![y_1]\!], \cdots, [\![y_s]\!]$ as inputs, where $s \leq \lfloor |N|/2|L| \rfloor$, $x_i, y_i \in (-2^{\ell}, 2^{\ell})$, and $i \in [1, s]$, and outputs a result group by multiplying the input ciphertext groups in pairs, i.e., $[\![x_1 y_1]\!], \cdots, [\![x_s y_s]\!]$. As shown in Algorithm 1, `BatchSMUL` consists of the following three steps:

1) S_1 chooses two groups of random numbers $r_{i,1}, r_{i,2} \xleftarrow{\$} \{0,1\}^{\sigma}$. Next, S_1 converts x_i and y_i into non-negative numbers and blinds them via the additive homomorphism as $[\![X_i]\!] \leftarrow [\![x_i]\!] \cdot [\![\delta]\!] \cdot [\![r_{i,1}]\!]$ and $[\![Y_i]\!] \leftarrow [\![y_i]\!] \cdot [\![\delta]\!] \cdot [\![r_{i,2}]\!]$. Then S_1 packs $[\![X_i]\!]$ and $[\![Y_i]\!]$ into one ciphertext by calculating
$$[\![c_i]\!] = [\![X_i]\!]^{L^{2i-1}} \cdot [\![Y_i]\!]^{L^{2i-2}}, \tag{6}$$
and packs a group of ciphertexts $[\![c_i]\!]$ into one ciphertext by computing $[\![C]\!] \leftarrow \prod_{i=1}^{i=s} [\![c_i]\!]$. Finally, S_1 partially decrypts $[\![C]\!]$ to get $[\![C_1]\!]$ and sends $\langle [\![C]\!], [\![C_1]\!] \rangle$ to S_2.
2) S_2 first partially decrypts $[\![C]\!]$ to get $[\![C_2]\!]$ and then obtains
$$\begin{aligned} C = {} & L^{2s-1}(x_s + r_{s,1} + \delta) + L^{2s-2}(y_s + r_{s,2} + \delta) \\ & + \cdots + L(x_1 + r_{1,1} + \delta) + (y_1 + r_{1,2} + \delta) \end{aligned} \tag{7}$$
via the threshold decryption. Next, S_2 unpacks C to obtain $x_i + r_{i,1}$ and $y_i + r_{i,2}$ by computing
$$\begin{cases} c_i = \lfloor C \mod L^{2i} / L^{2(i-1)} \rfloor, \\ x_i + r_{i,1} = \lfloor c_i / L \rfloor - \delta, \\ y_i + r_{i,2} = c_i \mod L - \delta, \end{cases} \tag{8}$$
Finally, S_2 encrypts $(x_i + r_{i,1}) \cdot (y_i + r_{i,2})$ and returns a group of ciphertexts $\{[\![(x_i + r_{i,1}) \cdot (y_i + r_{i,2})]\!]\}_{i=1}^{i=s}$ to S_1.
3) Since S_1 holds $r_{i,1}$, $r_{i,2}$, $[\![x_i]\!]$, and $[\![y_i]\!]$, it can locally compute $[\![-r_{i,2}(x_i + \delta)]\!]$, $[\![-r_{i,1}(y_i + \delta)]\!]$, $[\![-r_{i,1} \cdot r_{i,2}]\!]$, $[\![\delta(r_{i,1} + r_{i,2})]\!]$, where $i \in [1, s]$. Then, according to the additive homomorphism, S_1 computes $[\![(x_i + r_{i,1}) \cdot (y_i + r_{i,2})]\!] \cdot [\![-r_{i,2}(x_i + \delta)]\!] \cdot [\![-r_{i,1}(y_i + \delta)]\!] \cdot [\![-r_{i,1} \cdot r_{i,2}]\!] \cdot [\![\delta(r_{i,1} + r_{i,2})]\!]$ to obtain $[\![x_i y_i]\!]$ for $i \in \{1, 2, \ldots, s\}$.

Since it is more necessary to calculate the square of a ciphertext in our proposed design, we extend `BatchSMUL` to `BatchSquare`. Compared to `BatchSMUL`, `BatchSquare` is able to perform up to $\lfloor |N|/|L| \rfloor$ secure square calculations

Algorithm 1:
BatchSMUL($\langle [\![x_1]\!], \cdots, [\![x_s]\!] \rangle$, $\langle [\![y_1]\!], \cdots, [\![y_s]\!] \rangle$) $\rightarrow \langle [\![x_1 y_1]\!], \cdots, [\![x_s y_s]\!] \rangle$

Input: S_1 holds $\langle [\![x_1]\!], \cdots, [\![x_s]\!] \rangle$ and $\langle [\![y_1]\!], \cdots, [\![y_s]\!] \rangle$.
Output: S_1 obtains $\langle [\![x_1 y_1]\!], \cdots, [\![x_s y_s]\!] \rangle$.

▷ Step 1: S_1 computes

- $[\![X_i]\!] \leftarrow [\![x_i]\!] \cdot [\![\delta]\!] \cdot [\![r_{i,1}]\!]$ and $[\![Y_i]\!] \leftarrow [\![y_i]\!] \cdot [\![\delta]\!] \cdot [\![r_{i,2}]\!]$, where $i \in [1, s]$;
- $[\![c_i]\!] = [\![X_i]\!]^{L^{2i-1}} \cdot [\![Y_i]\!]^{L^{2i-2}}$;
- $[\![C]\!] \leftarrow \prod_{i=1}^{i=s} [\![c_i]\!]$ and $[\![C_1]\!] \leftarrow$ PDec$([\![C]\!])$;
- and sends $\langle [\![C]\!], [\![C_1]\!] \rangle$ to S_2.

▷ Step 2: S_2 computes

- $[\![C_2]\!] \leftarrow$ PDec$([\![C]\!])$ and $C \leftarrow$ TDec$([\![C_1]\!], [\![C_2]\!])$;
- $c_i = \lfloor C \bmod L^{2i} / L^{2(i-1)} \rfloor$, where $i \in [1, s]$;
- $x_i + r_{i,1} = \lfloor c_i / L \rfloor - \delta$ and $y_i + r_{i,2} = c_i \bmod L - \delta$, for $i \in [1, s]$;
- $[\![(x_i + r_{i,1}) \cdot (y_i + r_{i,2})]\!] \leftarrow$ Enc$((x_i + r_{i,1}) \cdot (y_i + r_{i,2}))$, for $i \in [1, s]$;
- and sends $\{[\![(x_i + r_{i,1}) \cdot (y_i + r_{i,2})]\!]\}_{i=1}^{i=s}$ to S_1.

▷ Step 3: For $i \in [1, s]$, S_1 computes

- $[\![-r_{i,2}(x_i + \delta)]\!] \leftarrow [\![x_i + \delta]\!]^{-r_{i,2}}$,
  $[\![-r_{i,1}(y_i + \delta)]\!] \leftarrow [\![y_i + \delta]\!]^{-r_{i,1}}$,
  $[\![-r_{i,1} \cdot r_{i,2}]\!] \leftarrow$ Enc$(-r_{i,1} \cdot r_{i,2})$,
  $[\![\delta(r_{i,1} + r_{i,2})]\!] \leftarrow$ Enc$(\delta(r_{i,1} + r_{i,2}))$;
- $[\![x_i y_i]\!] \leftarrow [\![(x_i + r_{i,1}) \cdot (y_i + r_{i,2})]\!] \cdot [\![-r_{i,2}(x_i + \delta)]\!] \cdot [\![-r_{i,1}(y_i + \delta)]\!] \cdot [\![-r_{i,1} \cdot r_{i,2}]\!] \cdot [\![\delta(r_{i,1} + r_{i,2})]\!]$.

at the same time. BatchSquare takes a group of ciphertexts $[\![x_1]\!], [\![x_2]\!], \cdots, [\![x_s]\!]$ as inputs, where $i \in [1, s]$, $s \leq \lfloor |N|/|L| \rfloor$, and $-2^\ell < x_i < 2^\ell$, and outputs the ciphertexts of the square of each input ciphertext, i.e., $[\![x_1^2]\!], \cdots, [\![x_s^2]\!]$. BatchSquare is proposed in Algorithm 2. The main difference between BatchSMUL and BatchSquare lies in the objects and the method of packing. At the step 1, S_1 packs $[\![X_i]\!]$ by computing

$$[\![C]\!] \leftarrow \prod_{i=1}^{i=s} [\![X_i]\!]^{L^{i-1}}. \tag{9}$$

At the step 2, S_2 unpacks C through $\lfloor C \bmod L^i / L^{i-1} \rfloor$, and then obtains $x_i + r_i$ by subtracting δ, where $i \in [1, s]$. At the step 3, S_1 obtains $[\![x_i^2]\!]$ by computing $[\![x_i^2]\!] \leftarrow [\![(x_i + r_i)^2]\!] \cdot [\![-2r_i(x_i + \delta)]\!] \cdot [\![-r_i^2]\!] \cdot [\![2\delta r_i]\!]$ for $i \in \{1, 2, \ldots, s\}$.

B. SMIN

Zhao *et al.* [31] have proposed a secure comparison protocol (SCMP). However, it is computationally expensive to extend SCMP to support the secure minimum protocol. Therefore, in this work, we propose a novel secure 2-minimum protocol and

Algorithm 2: BatchSquare($\langle [\![x_1]\!], \cdots, [\![x_s]\!] \rangle$) $\rightarrow \langle [\![x_1^2]\!], \cdots, [\![x_s^2]\!] \rangle$

Input: S_1 holds $\langle [\![x_1]\!], \cdots, [\![x_s]\!] \rangle$.
Output: S_1 obtains $\langle [\![x_1^2]\!], \cdots, [\![x_s^2]\!] \rangle$.

▷ Step 1: S_1 computes

- $[\![X_i]\!] \leftarrow [\![x_i]\!] \cdot [\![\delta]\!] \cdot [\![r_i]\!]$, where $i \in [1, s]$;
- $[\![C]\!] \leftarrow \prod_{i=1}^{i=s} [\![X_i]\!]^{L^{i-1}}$ and $[\![C_1]\!] \leftarrow$ PDec$([\![C]\!])$;
- and sends $\langle [\![C]\!], [\![C_1]\!] \rangle$ to S_2.

▷ Step 2: S_2 computes

- $[\![C_2]\!] \leftarrow$ PDec$([\![C]\!])$ and $C \leftarrow$ TDec$([\![C_1]\!], [\![C_2]\!])$;
- $x_i + r_i = \lfloor C \bmod L^i / L^{i-1} \rfloor - \delta$, where $i \in [1, s]$;
- $[\![(x_i + r_i)^2]\!] \leftarrow$ Enc$((x_i + r_i)^2)$, where $i \in [1, s]$;
- and sends $\{[\![(x_i + r_i)^2]\!]\}_{i=1}^{i=s}$ to S_1.

▷ Step 3: For $i \in [1, s]$, S_1 computes

- $[\![-2r_i(x_i + \delta)]\!] \leftarrow [\![x_i + \delta]\!]^{-2r_i}$,
  $[\![-r_i^2]\!] \leftarrow$ Enc$(-r_i^2)$,
  $[\![2\delta r_i]\!] \leftarrow$ Enc$(2\delta r_i)$;
- $[\![x_i^2]\!] \leftarrow [\![(x_i + r_i)^2]\!] \cdot [\![-2r_i(x_i + \delta)]\!] \cdot [\![-r_i^2]\!] \cdot [\![2\delta r_i]\!]$.

extend it to a secure n-minimum protocol. We first propose 2-SMIN that outputs the ciphertext of the minimum value of two given ciphertexts. Based on 2-SMIN, we develop n-SMIN that outputs the ciphertext of the minimum value of n given ciphertexts.

Given two ciphertexts $[\![x]\!]$ and $[\![y]\!]$, where $x, y \in [-2^\ell, 2^\ell]$, 2-SMIN outputs the ciphertext of the minimum value between x and y, i.e., 2-SMIN$([\![x]\!], [\![y]\!]) \rightarrow [\![min(x, y)]\!]$. 2-SMIN is shown in Algorithm 3, which consists of three steps as follows:

1) S_1 selects a random number $\pi \in \{0, 1\}$ and computes

$$[\![D]\!] = \begin{cases} ([\![x]\!] \cdot [\![y]\!]^{N-1})^{r_1} \cdot [\![r_1 + r_2]\!], & \pi = 0 \\ ([\![y]\!] \cdot [\![x]\!]^{N-1})^{r_1} \cdot [\![r_2]\!], & \pi = 1 \end{cases}, \tag{10}$$

where random numbers $r_1 \xleftarrow{\$} \{0, 1\}^\sigma \backslash \{0\}$ and r_2, s.t., $r_2 \leq \frac{N}{2}$ and $r_1 + r_2 > \frac{N}{2}$. σ is a secure parameter, e.g., $\sigma = 128$. Next, S_1 partially decrypts $[\![D]\!]$ to get $[\![D_1]\!]$ and then sends $\langle [\![D]\!], [\![D_1]\!], [\![x]\!], [\![y]\!] \rangle$ to S_2.

2) S_2 obtains D through partially decryption and threshold decryption with $[\![D]\!]$ and $[\![D_1]\!]$. Based on the value of D, S_2 assigns $[\![d_0]\!]$ with a refreshed $[\![x]\!]$ or $[\![y]\!]$. That is,

$$[\![d_0]\!] = \begin{cases} [\![y]\!] \cdot [\![0]\!]^r, & D > \frac{N}{2} \\ [\![x]\!] \cdot [\![0]\!]^r, & D \leq \frac{N}{2} \end{cases}, \tag{11}$$

where r is a random number used to refresh ciphertext, e.g., $r \xleftarrow{\$} \{0, 1\}^\ell$. S_2 returns $[\![d_0]\!]$ to S_1.

3) S_1 obtains the ciphertext of the minimum one between two ciphertexts according to the value of π generated in Step 1. If $\pi = 0$, S_1 sets $[\![min(x, y)]\!] = [\![d_0]\!]$, otherwise $(\pi = 1)$, S_1 computes $[\![min(x, y)]\!] = [\![x]\!] \cdot [\![y]\!] \cdot [\![d_0]\!]^{N-1}$.

Algorithm 3: 2-SMIN($[\![x]\!], [\![y]\!]$) $\rightarrow [\![min(x, y)]\!]$

Input: S_1 holds $[\![x]\!]$ and $[\![y]\!]$.
Output: S_1 obtains $[\![min(x, y)]\!]$.

▷ Step 1: S_1 computes

- $[\![D]\!] \leftarrow ([\![x]\!] \cdot [\![y]\!]^{N-1})^{r_1} \cdot [\![r_1 + r_2]\!]$ if $\pi = 0$,
or $[\![D]\!] \leftarrow ([\![y]\!] \cdot [\![x]\!]^{N-1})^{r_1} \cdot [\![r_2]\!]$ if $\pi = 1$,
where $r_1 \xleftarrow{\$} \{0, 1\}^{\sigma} \backslash \{0\}$, $r_2 \leq \frac{N}{2}$ and $r_1 + r_2 > \frac{N}{2}$;
- $[\![D_1]\!] \leftarrow$ PDec($[\![D]\!]$);
- and sends $\langle [\![D]\!], [\![D_1]\!], [\![x]\!], [\![y]\!] \rangle$ to S_2.

▷ Step 2: S_2 computes

- $[\![D_2]\!] \leftarrow$ PDec($[\![D]\!]$) and $D \leftarrow$ TDec($[\![D_1]\!], [\![D_2]\!]$);
- $[\![d_0]\!] \leftarrow [\![y]\!] \cdot [\![0]\!]^r$ if $D > \frac{N}{2}$,
or $[\![d_0]\!] \leftarrow [\![x]\!] \cdot [\![0]\!]^r$ if $D \leq \frac{N}{2}$,
where $r \xleftarrow{\$} \{0, 1\}^{\ell}$;
- and sends $[\![d_0]\!]$ to S_1.

▷ Step 3: S_1 computes

- $[\![min(x, y)]\!] = [\![d_0]\!]$ if $\pi = 0$,
otherwise $[\![min(x, y)]\!] = [\![x]\!] \cdot [\![y]\!] \cdot [\![d_0]\!]^{N-1}$ if $\pi = 1$.

As described in Algorithm 4, it is easy to construct n-SMIN based on the proposed 2-SMIN. Formally, n-SMIN takes n ciphertexts $[\![x_1]\!], [\![x_2]\!], \cdots, [\![x_n]\!]$ as inputs, and outputs $[\![min(x_1, x_2, \cdots, x_n)]\!]$.

Algorithm 4: n-SMIN($[\![x_1]\!], [\![x_2]\!], \cdots, [\![x_n]\!]$) $\rightarrow [\![min(x_1, x_2, \cdots, x_n)]\!]$

Input: S_1 holds $[\![x_1]\!], [\![x_2]\!], \cdots, [\![x_n]\!]$.
Output: S_1 obtains $[\![min(x_1, x_2, \cdots, x_n)]\!]$.

$[\![min(x_1, x_2, \cdots, x_n)]\!] \leftarrow [\![x_1]\!]$;
$t = 2$;
while $t \leq n$ **do**
  $[\![min(x_1, x_2, \cdots, x_n)]\!] \leftarrow$ 2-SMIN($[\![min(x_1, x_2, \cdots, x_n)]\!], [\![x_t]\!]$);
  $t = t + 1$;
return $[\![min(x_1, x_2, \cdots, x_n)]\!]$.

VI. Pura Design

In this section, we first formulate privacy-preserving face recognition. Next, we provide a brief overview of the proposed Pura. Following that, we elaborate on Pura.

A. Problem Formulation

Roughly speaking, privacy-preserving face recognition (PPFR) enables face recognition without leaking facial information. In this work, our proposed Pura fulfills PPFR through encrypting facial information, e.g., feature vectors.

Privacy-preserving face recognition can be formulated as:

$$\Gamma([\![p]\!], [\![\mathcal{D}]\!]) = \begin{cases} [\![ID]\!], & \Omega^{\diamond}([\![p]\!], [\![\mathcal{D}]\!]) \leq [\![\epsilon]\!] \\ \perp, & otherwise \end{cases}, \quad (12)$$

where $[\![p]\!]$ represents an encrypted feature vector to be recognized, $[\![\mathcal{D}]\!]$ refers to an encrypted feature vector database, and $[\![\epsilon]\!]$ denotes an encrypted threshold. PPFR takes input $[\![p]\!]$ and $[\![\mathcal{D}]\!]$ and securely calculates $\Omega^{\diamond}([\![p]\!], [\![\mathcal{D}]\!])$ to obtain the minimal distance between $[\![p]\!]$ and all encrypted feature vectors in $[\![\mathcal{D}]\!]$. In this work, the distance metric is calculated by the square of Euclidean distance.

Note that a comparison operation over the ciphertexts is performed between a threshold $[\![\epsilon]\!]$ and the minimum distance between $[\![p]\!]$ and $[\![\mathcal{D}]\!]$. If $\Omega(p, \mathcal{D}) \leq \epsilon$, where $\Omega(p, \mathcal{D})$ means the minimum square of Euclidean distance between p and $\mathcal{D}$, PPFR returns an encrypted identity, otherwise, returns $\perp$.

B. Overview

Our solution consists of two phases: **the registration phase** and **the recognition phase**. The workflow of our proposed Pura is sketched in Fig. 2.

During the registration phase, the organization encrypts a feature vector database and horizontally splits it into two parts, which are sent to S_1 and S_2, respectively. Additionally, each server performs offline operations in advance following the work [31].

During the recognition phase, the user sends an encrypted feature vector to S_1 and S_2, along with an encrypted random number specifically to S_1. Then, based on the proposed protocols and the parallel mechanism, S_1 and S_2 parallelly and jointly accomplish the square of Euclidean distance and minimum process. After that, S_1 masks the result, S_2 decrypts and returns the masked result to the user. Finally, the user recovers and obtains the recognition result.

C. Registration Phase

1) Data Transmission: During the registration phase, the organization needs to encrypt the feature vector database and store the encrypted feature vector database on the twin servers. As shown in Fig. 2, we divide the encrypted feature vector database into two parts and send them to two servers separately, instead of storing the encrypted feature vector database on a single server. Formally, the organization encrypts each value in the feature vector database $\mathcal{D}$, denoted as $[\![\mathcal{D}]\!] \leftarrow$ Enc($\mathcal{D}_{\Upsilon \times (n+1)}$) =

$$\begin{pmatrix} [\![ID_1]\!] & [\![v_{1,1}]\!] & [\![v_{1,2}]\!] & \cdots & [\![v_{1,n}]\!] \\ [\![ID_2]\!] & [\![v_{2,1}]\!] & [\![v_{2,2}]\!] & \cdots & [\![v_{2,n}]\!] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ [\![ID_{\Upsilon}]\!] & [\![v_{\Upsilon,1}]\!] & [\![v_{\Upsilon,2}]\!] & \cdots & [\![v_{\Upsilon,n}]\!] \end{pmatrix}.$$

In this work, the first $\lfloor \frac{\Upsilon}{2} \rfloor$ rows of $[\![\mathcal{D}]\!]$ are sent and stored on S_2, while the rest of $[\![\mathcal{D}]\!]$ are sent and stored on S_1. In this case, each server that holds half of the encrypted feature vector database can perform the protocols simultaneously, which fully takes advantage of the computation capability of each server.

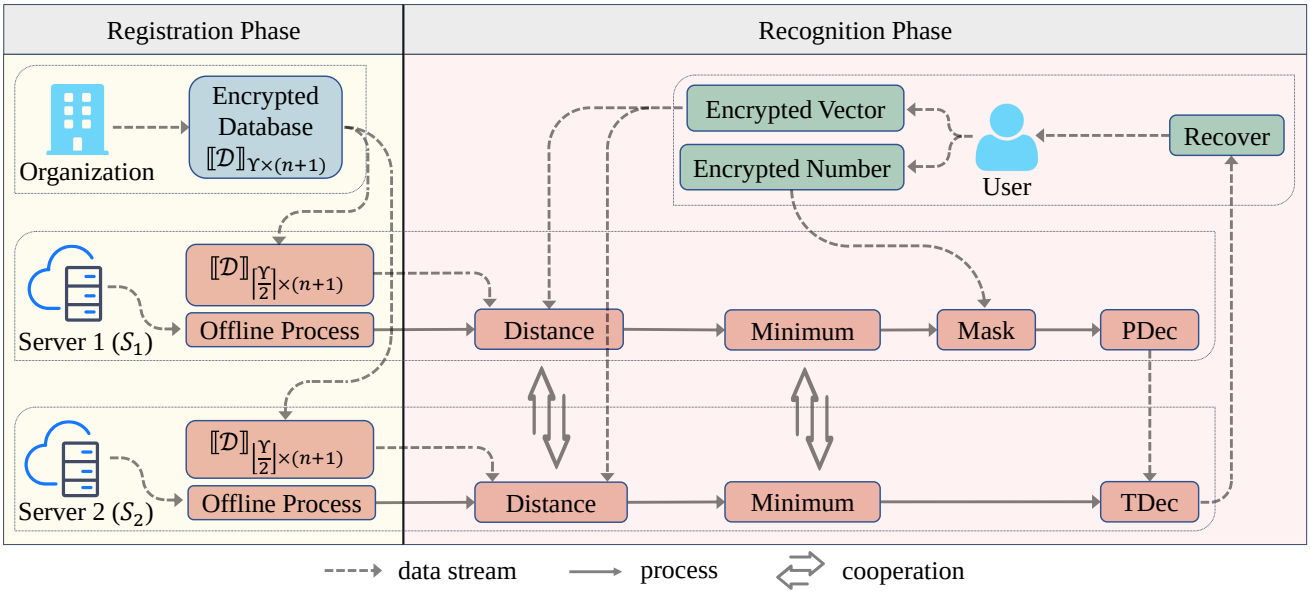


Fig. 2: System overview of Pura.

2) Offline Mechanism: Inspired by the offline mechanism in the study [31], during the registration phase, we also utilize the offline mechanism. Specifically, S_1 and S_2 respectively generate $[\![r_1]\!]$, $[\![r_2]\!]$, $[\![-r_1r_2]\!]$, $[\![\delta(r_1+r_2)]\!]$, $-r_1$, $-r_2$, L, $[\![r_1+r_2]\!]$, r, r_1, R, and $[\![R]\!]$, which can be directly used in the recognition phase. Note that the quantity of random numbers generated is determined by the demands of the recognition phase.

D. Recognition Phase

1) Square of Euclidean Distance: During the recognition phase, the user encrypts a probe feature vector ($[\![p]\!] \leftarrow \texttt{Enc}(p) = ([\![u_1]\!], [\![u_2]\!], \cdots, [\![u_n]\!])$) and then sends $[\![p]\!]$ to S_1 and S_2 at the same time. After obtaining the encrypted probe feature vector $[\![p]\!]$, S_1 and S_2 simultaneously compute the square of Euclidean distance between $[\![p]\!]$ and the item in $[\![\mathcal{D}]\!]$. Specifically, S_1 and S_2 first calculate the difference among feature vectors separately, which is formulated as

$$[\![Q]\!] = \begin{pmatrix} [\![u_1]\!] \cdot [\![v_{1,1}]\!]^{N-1} & \cdots & [\![u_n]\!] \cdot [\![v_{1,n}]\!]^{N-1} \\ \vdots & \ddots & \vdots \\ [\![u_1]\!] \cdot [\![v_{\phi,1}]\!]^{N-1} & \cdots & [\![u_n]\!] \cdot [\![v_{\phi,n}]\!]^{N-1} \end{pmatrix}, \tag{13}$$

where ϕ is the rows of $[\![\mathcal{D}]\!]$ that each server holds. After obtaining $[\![Q]\!]$, S_1 and S_2 parallelly and jointly perform `BatchSquare` to compute the square of every ciphertext in $[\![Q]\!]$. The result is denoted as a matrix $[\![E]\!]$ with ϕ rows and n columns of items $[\![e]\!]$. Once they finish `BatchSquare`, they can locally compute the square of Euclidean distances. According to the additive homomorphism, they compute the sum of each row in $[\![E]\!]$, which is defined as

$$[\![D_k]\!] = ([\![d_1]\!], \cdots, [\![d_\phi]\!]) = (\prod_{j=1}^{j=n} [\![e_{1,j}]\!], \cdots, \prod_{j=1}^{j=n} [\![e_{\phi,j}]\!]), \tag{14}$$

where S_1 and S_2 obtain $[\![D_1]\!]$ and $[\![D_2]\!]$, respectively.

2) Minimum Distance: (i) S_1 and S_2 jointly call n-`SMIN` and take input $[\![D_1]\!]$ and $[\![D_2]\!]$, respectively. Note that n of n-`SMIN` is the size of $[\![D_i]\!]$ for S_k, where $k \in \{1, 2\}$. After that, S_1 obtains the minimum result $[\![d_{S_1}]\!]$ among $[\![D_1]\!]$, and S_2 gets $[\![d_{S_2}]\!]$ in the same way. (ii) S_2 sends $[\![d_{S_2}]\!]$ to S_1. Then S_1 inputs two minimum results $[\![d_{S_1}]\!]$ and $[\![d_{S_2}]\!]$, as well as an encrypted threshold $[\![\epsilon]\!]$, and calls an extra n-`SMIN` protocol with S_2 to obtain the encrypted recognition result $[\![\gamma]\!]$. Specifically, n of the extra n-`SMIN` protocol is 3.

3) Mask and Recover: Upon obtaining $[\![\gamma]\!]$, S_1 masks $[\![\gamma]\!]$ with $[\![R]\!]$ via the additive homomorphism, i.e., $[\![\gamma+R]\!] = [\![\gamma]\!] \cdot [\![R]\!]$, where $R \xleftarrow{\$} \{0,1\}^\sigma$. Note that the random number R is generated and encrypted as $[\![R]\!]$ by the user. $[\![R]\!]$ is sent to S_1 along with the encrypted feature vector. Next, S_1 calls `PDec` to partially decrypt $[\![\gamma+R]\!]$ and sends the ciphertexts to S_2. S_2 obtains $\gamma+R$ by calling `PDec` and `TDec`. Eventually, the user receives $\gamma + R$ from S_2, and easily recovers the recognition result γ.

VII. Privacy Analysis

In this section, we demonstrate that the proposed secure computing protocols used to construct Pura prevent users' facial information from being leaked.

Due to the semantically secure property of threshold Paillier cryptosystems [29], [42], it has been widely adopted to protect users' images by performing operations in the encrypted domain. Zhao *et al.* [40] proved that $x + r$ is a chosen-plaintext attack (CPA) secure one-time key encryption scheme. According to the computational indistinguishability experiment between an adversary and a challenger presented in the work [43], we denote the computational indistinguishability experiment as $\text{PriR}^{cpa}_{\mathcal{A},x+r}$ and present a detailed description in the following.

1) An adversary $\mathcal{A}$ randomly chooses two random numbers x_0, x_1 such that $x_0, x_1 \in [-2^\ell, 2^\ell]$;

2) A challenger randomly selects $r \xleftarrow{\$} \{0,1\}^\sigma$ and $b \xleftarrow{\$} \{0,1\}$, where σ is a secure parameter. The challenger then calculates $x_b + r$ and returns to $\mathcal{A}$.
3) $\mathcal{A}$ yields a bit b'.
4) The experiment outputs 1 if $b' = b$; otherwise, it outputs 0. In the case when the output is 1, $\text{PriR}^{cpa}_{\mathcal{A},x+r}$ equals to 1, which means $\mathcal{A}$ succeeds.

***Theorem** 1:* For two random numbers $x_0, x_1 \in [-2^\ell, 2^\ell]$, $x_0 + r_0$ and $x_1 + r_1$ are considered computationally indistinguishable, where $r_0, r_1 \xleftarrow{\$} \{0,1\}^\sigma$. More precisely, given two random numbers $x_b \in [-2^\ell, 2^\ell]$ and $r \xleftarrow{\$} \{0,1\}^\sigma$, the formula $\Pr[b' = b | x_b + r] \leq \frac{1}{2} + \text{negl}(\sigma)$ always holds, where $\text{negl}(\sigma)$ is a negligible function of σ and $b, b' \xleftarrow{\$} \{0,1\}$. The study [40] offers the proof details.

***Corollary** 1:* Assuming a pair of ciphertexts of length s: $[\![x_1]\!], [\![x_2]\!], \cdots, [\![x_s]\!]$ and $[\![y_1]\!], [\![y_2]\!], \cdots, [\![y_s]\!]$, where $x_i, y_i \in [-2^\ell, 2^\ell]$, the proposed `BatchSMUL` protocol securely computes $[\![x_1 y_1]\!], [\![x_2 y_2]\!], \cdots, [\![x_s y_s]\!]$ under non-colluding attackers $\mathcal{A} = (\mathcal{A}_{S1}, \mathcal{A}_{S2})$, where $\mathcal{A}_{S1}$ and $\mathcal{A}_{S2}$ denote S_1 and S_2 as polynomial-time adversaries, respectively. That means `BatchSMUL` does not leak x_i and y_i to $\mathcal{A}_{S1}$ and $\mathcal{A}_{S2}$.

***Proof** 1:* We now construct the independent simulators $(\mathcal{S}_{S1}, \mathcal{S}_{S2})$, which simulates S_1 and S_2, respectively.

$\mathcal{S}_{S1}$ simulates the view of $\mathcal{A}_{S1}$ in the real in the following:

1) $\mathcal{S}_{S1}$ takes $\langle [\![x_i]\!], [\![y_i]\!], [\![(x_i + r_{i,1}) \cdot (y_i + r_{i,2})]\!] \rangle$ as inputs and then randomly chooses $\hat{x}_i, \hat{y}_i \in [-2^\ell, 2^\ell]$ and $\hat{r}_{i,1}, \hat{r}_{i,2}, \hat{\lambda}_1 \xleftarrow{\$} \{0,1\}^\sigma$.
2) $\mathcal{S}_{S1}$ calls `Enc` to encrypt $\hat{x}_i, \hat{y}_i, \hat{x}_i + \hat{r}_{i,1} + \delta, \hat{y}_i + \hat{r}_{i,2} + \delta, -\hat{r}_{i,2} \cdot (\hat{x}_i + \delta), -\hat{r}_{i,1} \cdot (\hat{y}_i + \delta), -\hat{r}_{i,1} \cdot \hat{r}_{i,2}, \delta \cdot (\hat{r}_{i,1} + \hat{r}_{i,2})$ into $[\![\hat{x}_i]\!], [\![\hat{y}_i]\!], [\![\hat{X}_i]\!], [\![\hat{Y}_i]\!], [\![-\hat{r}_{i,2} \cdot (\hat{x}_i + \delta)]\!], [\![-\hat{r}_{i,1} \cdot (\hat{y}_i + \delta)]\!], [\![-\hat{r}_{i,1} \cdot \hat{r}_{i,2}]\!], [\![\delta \cdot (\hat{r}_{i,1} + \hat{r}_{i,2})]\!]$, respectively. Then, $\mathcal{S}_{S1}$ calculates $[\![\hat{C}]\!] = \prod_{i=1}^{i=s} [\![\hat{X}_i]\!]^{L^{2i-1}} \cdot [\![\hat{Y}_i]\!]^{L^{2i-2}}$, and $[\![\hat{C}_1]\!] = [\![\hat{C}]\!]^{\hat{\lambda}_1}$ by calling `PDec`, and $[\![\hat{x}_i \cdot \hat{y}_i]\!] = [\![(x_i + r_{i,1}) \cdot (y_i + r_{i,2})]\!] \cdot [\![-\hat{r}_{i,2} \cdot (\hat{x}_i + \delta)]\!] \cdot [\![-\hat{r}_{i,1} \cdot (\hat{y}_i + \delta)]\!] \cdot [\![-\hat{r}_{i,1} \cdot \hat{r}_{i,2}]\!] \cdot [\![\delta \cdot (\hat{r}_{i,1} + \hat{r}_{i,2})]\!]$.
3) $\mathcal{S}_{S1}$ yields $\langle [\![\hat{x}_i]\!], [\![\hat{y}_i]\!], [\![\hat{X}_i]\!], [\![\hat{Y}_i]\!], [\![\hat{C}]\!], [\![\hat{C}_1]\!], [\![-\hat{r}_{i,2} \cdot (\hat{x}_i + \delta)]\!], [\![-\hat{r}_{i,1} \cdot (\hat{y}_i + \delta)]\!], [\![-\hat{r}_{i,1} \cdot \hat{r}_{i,2}]\!]$, and $[\![\delta \cdot (\hat{r}_{i,1} + \hat{r}_{i,2})]\!] \rangle$ as the simulation of $\mathcal{A}_{S1}$'s entire view.

$\mathcal{A}_{S1}$ only learns $[\![\hat{C}]\!]$ and $[\![\hat{C}_1]\!]$. Since the (2,2)-threshold Paillier cryptosystem is semantically secure, $\mathcal{A}_{S_1}$ cannot infer $[\![\hat{x}_i]\!], [\![\hat{y}_i]\!], [\![\hat{x}_i \cdot \hat{y}_i]\!]$ from $[\![\hat{C}]\!]$ and $[\![\hat{C}_1]\!]$. Thus, $[\![\hat{x}_i]\!]$ and $[\![x_i]\!]$, $[\![\hat{y}_i]\!]$ and $[\![y_i]\!]$, $[\![\hat{C}]\!]$ and $[\![C]\!]$, $[\![\hat{C}_1]\!]$ and $[\![C_1]\!]$ are computationally indistinguishable.

$\mathcal{S}_{S2}$ simulates the view of $\mathcal{A}_{S2}$ in the real in the following:

1) $\mathcal{S}_{S2}$ takes $\langle [\![\sum_{i=s}^{i=1} L^{2i-1} \cdot (x_i + r_{i,1} + \delta) + L^{2i-2} \cdot (y_i + r_{i,2} + \delta)]\!], [\![\sum_{i=s}^{i=1} L^{2i-1} \cdot (x_i + r_{i,1} + \delta) + L^{2i-2} \cdot (y_i + r_{i,2} + \delta)]\!]^{\lambda_1} \rangle$ as inputs and then randomly chooses $\bar{x}_i, \bar{y}_i \in [-2^\ell, 2^\ell]$ and $\bar{r}_{i,1}, \bar{r}_{i,2}, \bar{\lambda}_1, \bar{\lambda}_2 \xleftarrow{\$} \{0,1\}^\sigma$.
2) $\mathcal{S}_{S2}$ calls `Enc` to encrypt $\bar{x}_i + \bar{r}_{i,1}, \bar{y}_i + \bar{r}_{i,2}, (\bar{x}_i + \bar{r}_{i,1}) \cdot (\bar{y}_i + \bar{r}_{i,2})$ into $[\![\bar{X}_i]\!], [\![\bar{Y}_i]\!], [\![(\bar{x}_i + \bar{r}_{i,1}) \cdot (\bar{y}_i + \bar{r}_{i,2})]\!]$, respectively, and calculates $[\![\bar{X}_{i,1}]\!] = [\![\bar{X}_i]\!]^{\bar{\lambda}_1}, [\![\bar{X}_{i,2}]\!] = [\![\bar{X}_i]\!]^{\bar{\lambda}_2}, [\![\bar{Y}_{i,1}]\!] = [\![\bar{Y}_i]\!]^{\bar{\lambda}_1}, [\![\bar{Y}_{i,2}]\!] = [\![\bar{Y}_i]\!]^{\bar{\lambda}_2}$ by calling `PDec`.
3) $\mathcal{S}_{S2}$ yields $\langle [\![\bar{X}_i]\!], [\![\bar{X}_{i,1}]\!], [\![\bar{X}_{i,2}]\!], [\![\bar{Y}_i]\!], [\![\bar{Y}_{i,1}]\!], [\![\bar{Y}_{i,2}]\!], \bar{x}_i + \bar{r}_{i,1}, \bar{y}_i + \bar{r}_{i,2}, (\bar{x}_i + \bar{r}_{i,1}) \cdot (\bar{y}_i + \bar{r}_{i,2}), [\![(\bar{x}_i + \bar{r}_{i,1}) \cdot (\bar{y}_i + \bar{r}_{i,2})]\!] \rangle$ as the simulation of $\mathcal{A}_{S2}$'s entire view.

In the view of $\mathcal{A}_{S2}$, $\mathcal{A}_{S2}$ only learns $(\bar{x}_i + \bar{r}_{i,1}) \cdot (\bar{y}_i + \bar{r}_{i,2})$, so it fails to get $\bar{x}_i \cdot \bar{y}_i$. As the (2,2)-threshold Paillier cryptosystem is semantically secure, $[\![\bar{x}_i]\!]$ and $[\![x_i]\!]$, $[\![\bar{y}_i]\!]$ and $[\![y_i]\!]$ are computationally indistinguishable, demonstrating that $\mathcal{S}_{S2}$ in an ideal execution and $\mathcal{A}_{S2}$ in the real world are computationally indistinguishable.

Generally speaking, `BatchSMUL` is secure to calculate $[\![x_i \cdot y_i]\!]$ and does not leak x_i and y_i to $\mathcal{A}_{S1}$ and $\mathcal{A}_{S2}$.

***Corollary** 2:* Taking s ciphertexts $[\![x_1]\!], [\![x_2]\!], \cdots, [\![x_s]\!]$, where $-2^\ell \leq x_i \leq 2^\ell$, the proposed `BatchSquare` protocol securely calculates $[\![x_1^2]\!], [\![x_2^2]\!], \cdots, [\![x_s^2]\!]$ under non-colluding attackers $\mathcal{A} = (\mathcal{A}_{S1}, \mathcal{A}_{S2})$.

***Proof** 2:* Corollary 2 can easily be proved following the proof method of Corollary 1.

***Theorem** 2:* For two random numbers x_0, x_1 such that $x_0, x_1 \in [-2^\ell, 2^\ell]$, given $d = r_1 \cdot (x - y + 1) + r_2$ or $d = r_1 \cdot (y - x) + r_2$, where r_1, r_2 are two positive random numbers such that $r_1 \xleftarrow{\$} \{0,1\}^\sigma \backslash \{0\}, r_1 + r_2 > \frac{N}{2}$ and $r_2 \leq \frac{N}{2}$, if $\Pr[d = r_1 \cdot (x - y + 1) + r_2] = \Pr[d = r_1 \cdot (y - x) + r_2] = \frac{1}{2}$, the probability of successfully comparing the relative size of x and y is $\frac{1}{2}$. More formally, for random number $x, y \in [-2^\ell, 2^\ell]$, $\Pr[d > \frac{N}{2}] = \Pr[d \leq \frac{N}{2}] = \frac{1}{2}$. Detailed proof can be found in the study [40].

***Corollary** 3:* Given two ciphertexts $[\![x]\!]$ and $[\![y]\!]$ such that $x, y \in [-2^\ell, 2^\ell]$, the proposed 2-`SMIN` protocol securely computes $[\![min(x, y)]\!]$ under non-colluding attackers $\mathcal{A} = (\mathcal{A}_{S1}, \mathcal{A}_{S2})$.

***Proof** 3:* Following the proof method of Corollary 1, Corollary 3 is proved in the following.

1) In the view of $\mathcal{A}_{S1}$, $\mathcal{A}_{S1}$ only learns the ciphertexts $[\![x]\!], [\![y]\!]$, and $[\![r_1(x - y) + (r_1 + r_2)]\!]$ or $[\![r_1(y - x) + r_2]\!]$, so it fails to obtain x, y, and $min(x, y)$.
2) In the view of $\mathcal{A}_{S2}$, $\mathcal{A}_{S2}$ can learn $r_1(x - y) + (r_1 + r_2)$ or $r_1(y - x) + r_2$. However, according to Theorem 1 and Theorem 2, $\mathcal{A}_{S2}$ fails to obtain x, y and $min(x, y)$ from $r_1(x - y) + (r_1 + r_2)$ or $r_1(y - x) + r_2$.

In conclusion, 2-`SMIN` securely computes $[\![min(x, y)]\!]$ and does not leak x and y to $\mathcal{A}_{S1}$ and $\mathcal{A}_{S2}$.

***Corollary** 4:* Given n ciphertexts $[\![x_1]\!], [\![x_2]\!], \cdots, [\![x_n]\!]$, where $-2^\ell \leq x_i \leq 2^\ell$, the proposed n-`SMIN` protocol securely calculates $[\![min(x_1, x_2, \cdots, x_n)]\!]$ in a semi-honest (non-colluding) model.

***Proof** 4:* Based on Corollary 3, it is easy to derive Corollary 4, since n-`SMIN` is simply calling 2-`SMIN` repeatedly.

***Theorem** 3:* Given an encrypted feature vector $[\![p]\!]$ and an encrypted feature vector database $[\![\mathcal{D}]\!]$, S_1 and S_2 cannot obtain the raw data p and $\mathcal{D}$.

***Proof** 5:* The proposed Pura consists of the following operations: compute the difference between an encrypted input feature vector and an encrypted feature vector database via homomorphic subtraction; calculate the square of every difference by calling `BatchSquare`; compute the square of Euclidean distance through homomorphic addition; search for the minimum distance and compare it with a threshold by calling n-`SMIN`.

According to the threshold Paillier cryptosystem [29], the addition and subtraction operations over ciphertexts do not

leak plaintext data. According to Corollary 2, S_1 and S_2 can safely cooperate to compute the square of a pair of ciphertexts by calling `BatchSquare`. Based on Corollary 4, n-`SMIN` can perform the minimum operation among a group of ciphertexts without leaking privacy. Lastly, the encrypted recognition result is masked by S_1 and the user recovers the real result. Based on the assumption that S_1 and S_2 do not collude with each other, neither S_1 nor S_2 is able to learn the plaintext of the recognition result, which can only be obtained by the user.

In conclusion, our proposed Pura solution can securely perform the PPFR task without revealing the original data to $\mathcal{A}_{S2}$ and $\mathcal{A}_{S2}$. In other words, Pura protects users' facial information and avoids private data leakage.

Finally, it can be easily demonstrated that Pura satisfies the three security requirements of ISO/IEC 24745 [44]: irreversibility, unlinkability, and confidentiality. First, the Paillier homomorphic cryptosystem ensures irreversibility, as attackers cannot retrieve the original plaintext data from leaked encrypted data. Second, the unlinkability is guaranteed by using different public/private key pairs for distinct face recognition systems, preventing the association of samples from the same user across different systems. Third, according to Theorem 3, our scheme meets the confidentiality requirement of ISO/IEC 24745, ensuring that neither external adversaries nor the internal servers can obtain users' facial data.

VIII. Experiments

In this section, we evaluate the effectiveness and efficiency of our proposed Pura.

A. Experimental setting

We adopt the popular Labeled Faces in the Wild (LFW) [45] with 13,233 images of 5,749 persons and CASIA-WebFace (WebFace) [46] with 459,737 images of 10,575 persons as our experimental datasets. We transform each image into a 512-dimensional feature vector in advance using a pre-trained deep-learning model Facenet [33]. Each value of the feature vector is mapped into the range of $[0, 1]$. Since the values of the feature vectors are real numbers, we transform the real numbers into integers by multiplying a constant number 10,000. Our scheme is implemented in C++, and the experiments are conducted on two servers, each of which is equipped with an AMD EPYC 7402 CPU and 128 GB of RAM.

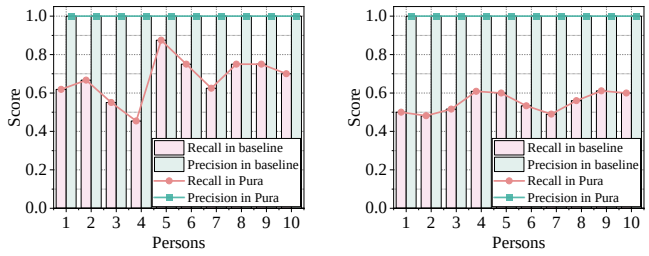
Our proposed secure computing protocols are based on SOCI$^+$ [31]. We also implement the protocols using SOCI [40], called Naive. We conduct the experiments setting N as 1024. We compare Pura with two related solutions [23] and [28] based on BFV [47]. Note that the work [23] adopts a row-wise strategy and the scheme [28] utilizes a column-wise strategy. We conduct the experiments at an 80-bit security level. We set $prime\_modulus = 265,774,897$ and $hensel\_lifting = 1$. The parameter $slots$ is to control the number of plaintext encrypted in a single ciphertext. We select $slots = 520$ for the work [23], and $slots = 1,040$ for the scheme [28]. Other parameters are shown in Table I.

TABLE I: Parameter settings of different $slots$.

$slots$	$cyclotomic\_polynomial$	$modulus\_chain\_bits$
520	4,847	119
1040	11,135	230

B. Effectiveness

To evaluate the effectiveness of our solution, we compare it with a baseline face recognition algorithm, which is formulated in Section III-B. We randomly choose 200 images of 10 people in LFW and 300 images of 10 people in WebFace. All images are transformed into 512-dimensional feature vectors in advance. For each vector, we exclude it from the feature vector database and select it as the probe feature vector to be identified. We compare the precision and recall between baseline and Pura, where the precision and recall are formulated by $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$, where TP, FP and FN represent $true\ positive$, $false\ positive$ and $false\ negative$, respectively. As the comparison result shown in Fig. 3, it is obvious that Pura has the same precision and recall with baseline, which demonstrates the effectiveness of our proposed Pura.



(a) LFW (b) WebFace

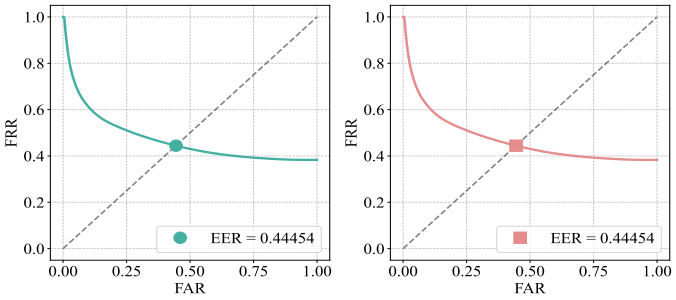
Fig. 3: Feasibility evaluation on LFW and WebFace.

We conduct a statistical analysis employing the Wilcoxon rank-sum test at a significance level of 0.05. When the p-value of the Wilcoxon rank-sum test is less than 0.05, it indicates that there are significant differences between the two solutions. Conversely, if the p-value exceeds 0.05, the test indicates no significant difference between the two solutions. As seen from Table II, the statistical p-value between Pura and baseline is greater than 0.05 in terms of the sum, mean, and variance. Therefore, we conclude that there is no significant difference between Pura and baseline when performing face recognition. Hence, we conclude that Pura is a feasible privacy-preserving face recognition solution.

TABLE II: Statistical analysis between Pura and baseline.

Dataset	Sum	Mean	Variance
LFW	1	1	1
WebFace	1	1	1

We further compare the Detection Error Tradeoff (DET) curves of Pura and baseline. The Equal Error Rate (EER) is the point where the False Acceptance Rate (FAR) equals the



(a) Pura (b) Basseline

Fig. 4: DET curve comparison of Pura with baseline.

False Rejection Rate (FRR), presenting the performance of the face recognition system. As shown in Fig. 4, the EER for Pura is 0.44454, which matches the EER for baseline. This result indicates that the proposed encryption scheme has no affect on the performance of the face recognition system, further demonstrating the effectiveness of Pura.

C. Efficiency

We first compare the encrypted feature vector database storage. We compare Pura with the work [23] and the scheme [28]. The result is depicted in Fig. 5. It can be observed that BFV-based schemes suffer from high storage overhead. In terms of the storage overhead of a single server, the solution [28] consumes more than 6 GB, while the solution [23] costs over 3 GB when the dataset size is 10,000. In contrast to the solutions [23] and [28], our proposed Pura consumes around 1 GB of storage overhead on a single server. Consequently, we can say that the proposed Pura enjoys less storage cost.

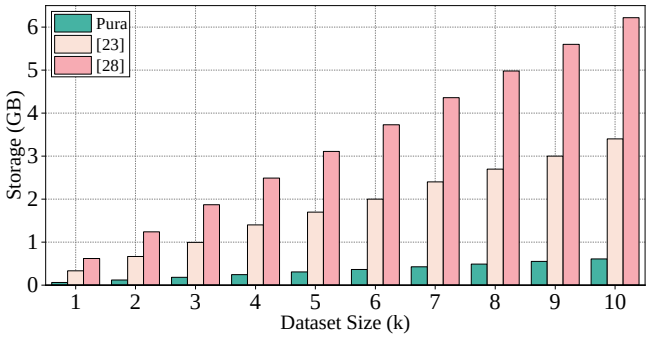
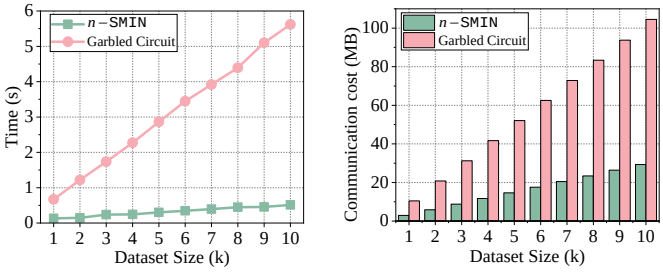


Fig. 5: Comparison of storage costs of a single server among three solutions.

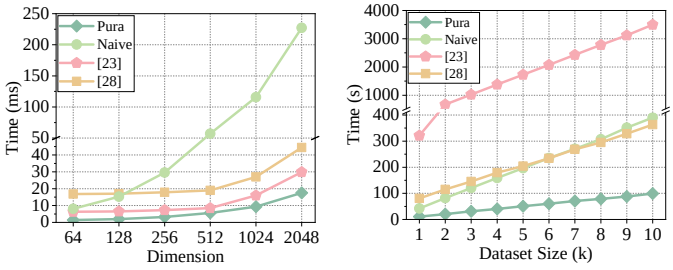
Both solutions [23] and [28] adopt BFV to perform the square operation. In order to demonstrate the efficiency of the proposed `BatchSquare` protocol, we compare the runtime of `BatchSquare` and the BFV-based scheme, as well as `BatchSMUL`, in terms of square operations. We use the HElib library [48] to implement the BFV square operation, named `BFV-Square`. We range the data size from 1,000 to 10,000 to compare the runtime. As shown in Table III, `BatchSquare` takes less time than `BFV-Square` generally. Note that in the square operation, `BatchSquare` can reduce the redundancies in `BatchSMUL`, thereby decreasing the number of computed expressions and increasing the number of batch computations. Consequently, `BatchSquare` outperforms `BatchSMUL` in the square operation. Additionally, the growth rates of the runtime for `BFV-Square`, `BatchSMUL`, and `BatchSquare` are 55.47, 41.15, and 25.98, respectively. Therefore, as the size of the dataset grows, `BatchSquare` shows significant advantage in terms of runtime.

Since the garbled circuit is one of the most popular approaches to finding the minimum value securely, we extensively compare our proposed n-`SMIN` scheme with the garbled circuit. We set the same security level (i.e., 80-bit security) in this experiment. Following the approach of constructing a garbled circuit to obtain the minimum value in the work [28], we separate each value x into two shares x_1 and x_2, such that $x = x_1 + x_2$, and store x_1 and x_2 on different servers. In the n-`SMIN` scheme, each value is encrypted into ciphertext, and all ciphertexts are stored separately on different servers. As shown in Fig. 6, n-`SMIN` requires less runtime and communication cost to find the minimum value than the garbled circuit. All in all, n-`SMIN` significantly outperforms the traditional garbled circuit.



(a) Runtime (b) Communication cost

Fig. 6: Comparison between n-`SMIN` and garbled circuit.



(a) Feature vector encryption (b) Face recognition

Fig. 7: Runtime comparison among four solutions.

We evaluate the efficiency of computation for Pura through runtime. Fig. 7 shows the runtime comparison between Pura, Naive, solutions [23] and [28]. Fig. 7(a) shows the runtime of encrypting a feature vector with varying dimensions. It can be observed that our proposed Pura outperforms the other schemes. Among the four schemes, Pura shows the slowest increase in runtime as the dimension of the feature vector

TABLE III: Runtime of the square operation.

Dataset Size (k)	1	2	3	4	5	6	7	8	9	10
BFV-Square	57 ms	112 ms	170 ms	226 ms	279 ms	333 ms	391 ms	446 ms	501 ms	557 ms
BatchSMUL	104 ms	156 ms	191 ms	238 ms	274 ms	321 ms	354 ms	395 ms	439 ms	481 ms
BatchSquare	60 ms	90 ms	125 ms	153 ms	173 ms	202 ms	224 ms	240 ms	272 ms	304 ms

TABLE IV: Comparison of communication cost (GB) among three solutions.

Dataset Size (k)	1	2	3	4	5	6	7	8	9	10
Pura	0.2061	0.4121	0.6172	0.8229	1.0295	1.2344	1.4404	1.6463	1.8516	2.0572
[23]	0.0012	0.0021	0.0030	0.0038	0.0047	0.0056	0.0064	0.0073	0.0084	0.0090
[28]	1.6777	1.6872	1.6966	1.7062	1.7158	1.7251	1.7347	1.7442	1.7538	1.7631

changes. From Fig. 7(b), we see that Pura is significantly faster than the other algorithms. In contrast to the work [28], Pura is 16 times faster than the work [28] when the dataset size is 1,000. And the runtime of Pura is twice that of the work [28] when the dataset size is 10,000. Since the work [23] utilizes a high-complexity strategy and excessive use of expensive homomorphic operations such as circular shifts, the recognition runtime is unacceptable. Thanks to the offline mechanism and the optimization of SOCI [40], Pura can save about 75% of runtime compared to Naive.

Table IV shows the comparison results in terms of communication cost among the three solutions. As indicated in Table IV, the communication cost of Pura is less than that of the work [28] in general. It can be observed that when the dataset size exceeds 9,000, the communication cost of Pura surpasses that of the work [28]. We consider that this trade-off between communication cost and runtime in our scheme is acceptable, according to Fig. 7(b). More crucially, the user experience is of greater importance for a face recognition system. Additionally, combining Table IV and Fig. 7(b), although the communication cost of scheme [23] is the lowest, the sacrificed computational efficiency is unacceptable. The scheme requires a single server to perform all the complex and costly computations. While this reduces the communication volume, it results in an intolerable recognition latency. Additionally, as the scheme allows the server to decrypt and access the plaintext, its security level is significantly lower than that of our proposed Pura.

To summarize, our proposed Pura excels in runtime efficiency, significantly outperforming the other schemes. Also, Pura achieves a good balance between communication cost and runtime.

IX. Conclusion

In this paper, we propose Pura, an efficient privacy-preserving face recognition solution, which enables the twin servers to efficiently identify a user without leaking privacy. To achieve non-interactive and privacy-preserving face recognition, we propose a novel PPFR framework based on the threshold Paillier cryptosystem. We carefully design several secure underlying computing protocols to support secure and efficient operations over ciphertexts, including multiplication, square, and minimum. Moreover, we utilize a parallel computing mechanism to improve the efficiency of privacy-preserving face recognition. Our proposed Pura avoids degrading recognition accuracy while achieving remarkable performance. The privacy analysis details the security of our design through the proposed underlying protocols. The experimental results demonstrate that our solution outperforms the state-of-the-art in terms of communication cost and runtime. For future work, we intend to deploy our proposed Pura into a real product.

X. Acknowledgments

The authors would like to thank the Editor-in-Chief, the Associate Editor, and the reviewers for their valuable comments and suggestions. This work is partially supported by the National Natural Science Foundation of China (No. 62202358), the China PostDoctoral Science Foundation (No. 2023TQ0258), the Guangdong Natural Science and Foundation (No. 2024A1515011039).

References

[1] V. Manjula, L. Baboo *et al.*, "Face detection identification and tracking by prdit algorithm using image database for crime investigation," *Int. J. Comput. Appl.*, vol. 38, no. 10, pp. 40–46, 2012.

[2] K. Lander, V. Bruce, and M. Bindemann, "Use-inspired basic research on individual differences in face identification: Implications for criminal investigation and security," *Cognitive Research: Principles and Implications*, vol. 3, no. 1, pp. 1–13, 2018.

[3] T. Pinthong, W. Yimyam, N. Chumuang, and M. Ketcham, "Face recognition system for financial identity theft protection," in *2020 15th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*. IEEE, 2020, pp. 1–6.

[4] S. Haji and A. Varol, "Real time face recognition system (rtfrs)," in *2016 4th International Symposium on Digital Forensic and Security (ISDFS)*. IEEE, 2016, pp. 107–111.

[5] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[6] V. Perlibakas, "Distance measures for pca-based face recognition," *Pattern recognition letters*, vol. 25, no. 6, pp. 711–724, 2004.

[7] N. Muslim and S. Islam, "Face recognition in the edge cloud," in *Proceedings of the International Conference on Imaging, Signal Processing and Communication*, 2017, pp. 5–9.

[8] A. A. Pawle and V. P. Pawar, "Face recognition system (frs) on cloud computing for user authentication," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 3, no. 4, pp. 189–192, 2013.

[9] S. Siregar, M. Syahputra, and R. Rahmat, "Human face recognition using eigenface in cloud computing environment," in *IOP conference series: materials science and engineering*, vol. 308, no. 1. IOP Publishing, 2018, p. 012013.

[10] L. Laishram, M. Shaheryar, J. T. Lee, and S. K. Jung, "Toward a privacy-preserving face recognition system: A survey of leakages and solutions," *ACM Computing Surveys*, 2024.

[11] R. Fallon, "Celebgate: Two methodological approaches to the 2014 celebrity photo hacks," in *Internet Science: Second International Conference, INSCI 2015, Brussels, Belgium, May 27-29, 2015, Proceedings 2*. Springer, 2015, pp. 49–60.

[12] C.-A. Staicu and M. Pradel, "Leaky images: Targeted privacy attacks in the web," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 923–939.
[13] M. Ogburn, C. Turner, and P. Dahal, "Homomorphic encryption," *Procedia Computer Science*, vol. 20, pp. 502–509, 2013.
[14] A. Beimel, "Secret-sharing schemes: A survey," in *International conference on coding and cryptology*. Springer, 2011, pp. 11–46.
[15] X. Xiong, S. Liu, D. Li, Z. Cai, and X. Niu, "A comprehensive survey on local differential privacy," *Security and Communication Networks*, vol. 2020, no. 1, p. 8829523, 2020.
[16] Z. Qin, J. Yan, K. Ren, C. W. Chen, and C. Wang, "Towards efficient privacy-preserving image feature extraction in cloud computing," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 497–506.
[17] S. Guo, T. Xiang, and X. Li, "Towards efficient privacy-preserving face recognition in the cloud," *Signal Processing*, vol. 164, pp. 320–328, 2019.
[18] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Privacy Enhancing Technologies: 9th International Symposium, PETS 2009, Seattle, WA, USA, August 5-7, 2009. Proceedings 9*. Springer, 2009, pp. 235–253.
[19] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *International conference on information security and cryptology*. Springer, 2009, pp. 229–244.
[20] J. Lei, Q. Pei, Y. Wang, W. Sun, and X. Liu, "Privface: Fast privacy-preserving face authentication with revocable and reusable biometric credentials," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3101–3112, 2021.
[21] D. Evans, Y. Huang, J. Katz, and L. Malka, "Efficient privacy-preserving biometric identification," in *Proceedings of the 17th conference Network and Distributed System Security Symposium, NDSS*, vol. 68, 2011, pp. 90–98.
[22] C. Xiang, C. Tang, Y. Cai, and Q. Xu, "Privacy-preserving face recognition with outsourced computation," *Soft Computing*, vol. 20, pp. 3735–3744, 2016.
[23] P. Drozdowski, N. Buchmann, C. Rathgeb, M. Margraf, and C. Busch, "On the application of homomorphic encryption to face identification," in *2019 International Conference of the Biometrics Special Interest Group (BIOSIG)*. IEEE, 2019, pp. 1–5.
[24] M. Blanton and M. Aliasgari, "Secure outsourced computation of iris matching," *Journal of Computer Security*, vol. 20, no. 2-3, pp. 259–305, 2012.
[25] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe, "Privacy preserving face recognition utilizing differential privacy," *Computers & Security*, vol. 97, p. 101951, 2020.
[26] L. Ou, Y. He, S. Liao, Z. Qin, Y. Hong, D. Zhang, and X. Jia, "Faceidp: Face identification differential privacy via dictionary learning neural networks," *IEEE Access*, vol. 11, pp. 31 829–31 841, 2023.
[27] H. Chun, Y. Elmehdwi, F. Li, P. Bhattacharya, and W. Jiang, "Outsourceable two-party privacy-preserving biometric authentication," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*, 2014, pp. 401–412.
[28] H. Huang and L. Wang, "Efficient privacy-preserving face identification protocol," *IEEE Transactions on Services Computing*, vol. 16, no. 4, pp. 2632–2641, 2023.
[29] A. Lysyanskaya and C. Peikert, "Adaptive security in the threshold setting: From cryptosystems to signature schemes," in *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*. Springer, 2001, pp. 331–350.
[30] N. H. Barnouti, S. S. M. Al-Dabbagh, and W. E. Matti, "Face recognition: A literature review," *International Journal of Applied Information Systems*, vol. 11, no. 4, pp. 21–31, 2016.
[31] B. Zhao, W. Deng, X. Li, X. Liu, Q. Pei, and R. H. Deng, "SOCIˆ+: An enhanced toolkit for secure outsourcedcomputation on integers," *arXiv preprint arXiv:2309.15406*, 2023.
[32] V. Cherepanova, S. Reich, S. Dooley, H. Souri, J. Dickerson, M. Goldblum, and T. Goldstein, "A deep dive into dataset imbalance and bias in face identification," in *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, 2023, pp. 229–247.
[33] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.
[34] S. Kar, S. Hiremath, D. G. Joshi, V. K. Chadda, and A. Bajpai, "A multi-algorithmic face recognition system," in *2006 International Conference on Advanced Computing and Communications*. IEEE, 2006, pp. 321–326.
[35] D. N. Parmar and B. B. Mehta, "Face recognition methods & applications," *arXiv preprint arXiv:1403.0485*, 2014.
[36] M. P. Gawande and D. G. Agrawal, "Face recognition using pca and different distance classifiers," *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*, vol. 9, no. 1, pp. 1–5, 2014.
[37] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE transactions on information forensics and security*, vol. 7, no. 3, pp. 1053–1066, 2012.
[38] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 2014, pp. 664–675.
[39] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 19–38.
[40] B. Zhao, J. Yuan, X. Liu, Y. Wu, H. H. Pang, and R. H. Deng, "SOCI: A toolkit for secure outsourced computation on integers," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3637–3648, 2022.
[41] B. Zhao, Y. Li, X. Liu, H. H. Pang, and R. H. Deng, "FREED: An efficient privacy-preserving solution for person re-identification," in *2022 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2022, pp. 1–8.
[42] X. Liu, R. H. Deng, W. Ding, R. Lu, and B. Qin, "Privacy-preserving outsourced calculation on floating point numbers," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2513–2527, 2016.
[43] J. Katz and Y. Lindell, *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2007.
[44] K. ISO/IEC JTC1 SC27 Security Techniques, "Iso/iec 24745: 2011. information technology-security techniques-biometric information protection," 2011.
[45] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.
[46] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.
[47] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.
[48] S. Halevi and V. Shoup, "Algorithms in helib," in *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*. Springer, 2014, pp. 554–571.

Guotao Xu received the B.S. degree in Information Security from Guangdong University of Technology, Guangzhou, China, in 2023. He is currently working toward the M.S. degree in Guangzhou Institute of Technology, Xidian University, Guangzhou, China. His current research interest is privacy-preserving computation.

Bowen Zhao (Member, IEEE) received the Ph.D. degree in cyberspace security from the South China University of Technology, China, in 2020. He was a Research Scientist with the School of Computing and Information Systems, Singapore Management University, from 2020 to 2021. He is currently an Associate Professor with the Guangzhou Institute of Technology, Xidian University, Guangzhou, China. His current research interests include privacy-preserving computation and learning and privacy-preserving crowdsensing.

Yang Xiao (Member, IEEE) received the B.S. and Ph.D. degrees in communication engineering from Xidian University, Xi'an, China, in 2013 and 2020, respectively. From 2017 to 2019, he was supported by the China Scholarship Council to be a visiting Ph.D. student with the University of New South Wales, Sydney, NSW, Australia. He is currently a Lecturer with the State Key Laboratory of Integrated Services Networks, School of Cyber Engineering, Xidian University. His research interests include social networks, joint recommendations, graph neural networks, trust evaluation, and blockchain.

PLACE PHOTO HERE

Yantao Zhong, China Resources Intelligent Computing Technology (Guangdong) Co., Ltd., Senior Engineer, 1980-, from Shangrao, Jiangxi, focusing on cryptography and privacy computing technology.

PLACE PHOTO HERE

Liang Zhai, Chinese Academy of Surveying and Mapping, Beijing, China.

Qingqi Pei (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science and cryptography from Xidian University, Xi'an, China, in 1998, 2005, and 2008, respectively. He is currently a Professor and a member of the State Key Laboratory of Integrated Services Networks, also a Professional Member of ACM, and a Senior Member of the Chinese Institute of Electronics and China Computer Federation. His research interests focus on digital contents protection and wireless networks and security.