PBio: Enabling Cross-organizational Biometric Authentication Service through Secure Sharing of Biometric Templates

Jia-Chng Loh*, Geong-Sen Poh†, Jason H. M. Ying*, Jia Xu†, Hoon Wei Lim†
NUS-Singtel Cyber Security Lab
{*{dcsljc,dcsyhmj}@nus.edu.sg, †{geongsen.poh,jia.xu,hoonwei.lim}@trustwave.com

Jonathan Pan, Weiyang Wong
Home Team Science & Technology Agency
{jonathan_pan,wong_weiyang}@htx.gov.sg

Abstract
Prior works in privacy-preserving biometric authentication mostly focus on the following setting. An organization collects users’ biometric data during registration and later authorized access to the organization services after successful authentication. Each organization has to maintain its own biometric database. Similarly each user has to release her biometric information to multiple organizations; Independently, government authorities are making their extensive, nation-wide biometric database available to agencies and organizations, for countries that allow such access. This will enable organizations to provide authentication without maintaining biometric databases, while users only need to register once. However privacy remains a concern. We propose a privacy-preserving system, PBio, for this new setting. The core component of PBio is a new protocol comprising distance recoverable encryption and secure distance computation. We introduce an encrypt-then-split mechanism such that each of the organizations holds only an encrypted partial biometric database. This minimizes the risk of template reconstruction in the event that the encrypted partial database is recovered due to leak of the encryption key. PBio is also secure even when the organizations collude. A by-product benefit is that the use of encrypted partial templates allows quicker rejection for non-matching instances. We implemented a cloud-based prototype with desktop and Android applications. Our experiment results based on real remote users show that PBio is highly efficient. A round-trip authentication takes approximately 74ms (desktop) and 626ms (Android). The computation and communication overhead introduced by our new cryptographic protocol is only about 10ms (desktop) and 54ms (Android).

1 Introduction
The use of biometric information, such as fingerprint, face and iris, for authentication purposes has proliferated in recent years. Allowing various organizations, such as banks and government agencies, access to biometric templates hosted by a central and trusted entity will be advantageous in multiple aspects. First it allows organizations that currently have no access to have direct access to a readily available database. Second, these organizations do not need to invest in the infrastructure required to enroll new users and store raw biometric databases of their own. This also reduces the risk of potential data breaches. Third, a user only needs to register once with the trusted entity to access to services provided by various organizations. Singapore’s SingPass face verification [16] and India’s Aadhaar project [59] are two recent examples, where organizations subscribe to the national biometric databases to enable authentication services to their client.

However, allowing organizations direct access to a central biometric database still needs to be handled with extra care. BioStar 2, a biometric security platform that provides biometric database hosting service, experienced a data breach incident resulting in the leakage of users’ biometric information [22]. Also, there are biometric practitioners that regard the use of biometric templates without any further protection as adequate as far as security and privacy is concerned. That is, a template should not contain sufficient information for reconstruction of the corresponding original biometric data [26]. On the contrary, significant progress has been made lately in the domain of biometric template reconstruction, see for example [9, 10, 23, 25, 47]. This is a major privacy concern since the reconstructed biometric templates can be used to identify or impersonate an individual. A more viable approach is to share a subset of the original database in a privacy-preserving manner. This way, the raw biometric templates remain with only the trusted entity and are isolated from the other organizations. It enables other organizations to authenticate the said individuals, but in such a way that the organizations cannot learn any biometric information from the shared database.

Use Case. We envision a biometric authentication service that allows an individual to simply scan her face at a kiosk, or capture her facial image through a mobile application (which will enable access even when the individual is abroad) to initiate a transaction with a bank or a government agency. The captured facial features will be matched against a shared
encrypted biometric database that is derived from a national biometric database. The use of biometrics as an alternative to conventional text-based passwords is of high interest to various smart nation initiatives worldwide, for example [74]. Specifically, a face verification application was recently proposed in collaboration between a Singapore-based bank, DBS, and a Singapore government agency, GovTech, to pilot face verification for easier digital banking sign-ups and to secure higher value transactions [16]. In the case of hotel check-in services, the physical presence of a reception staff is not required to perform this role. Instead, self check-in can be completed either through an on-site kiosk located within the premise or via mobile authentication [61, 86].

Existing Approaches. Current privacy-preserving biometric authentication schemes, see for example [5, 7, 21, 45, 67, 89], performed authentication directly between a user and a service provider (e.g. bank). There are also proposals that enable an organization to outsource the resources required, both computation and storage, for biometric authentication to a cloud service [13, 27, 29, 75, 78, 81, 87, 88]. In both settings, a user provides her biometric data to her service provider during registration. Thus, multiple registrations are required if the user intends to register with several service providers. This also means each organization collects and maintains its own set of biometric templates.

Our Approach. In our proposed system, PBio, a user only needs to register once with the entity responsible for a national-level biometric database to have access to a range of services provided through multiple service providers and agencies. There are a couple of key technical challenges in the design of PBio. First, users’ biometric information must be disseminated to the respective service providers in a secure and privacy-preserving manner. Second, the time required to complete an authentication check should be fast and comparable to that of existing biometric authentication systems. Third, the risk of reconstruction of biometric features should be minimal even in the event that the biometric database used by a service provider is leaked.

Our Contributions. To the best of our knowledge, PBio is the first system that allows cross-organizational and privacy-preserving biometric data sharing and authentication. Our contributions are as follows:

- We propose a provably secure protocol using lightweight distance recoverable encryption with orthogonal matrix and secure two-party computation scheme, GSHADE. This allows us to outsource the encrypted templates and securely authenticate users from different organizations without decryption.
- Our protocol makes use of a new encrypt-then-split construct. Encrypted biometric templates are split into two (or more) parts, where one part is given to a cloud service provider and the other parts to organizations that provide authentication services. A biometric feature set captured in real time can be tested against the encrypted partial biometric templates stored by an organization. The advantages of splitting the biometric template is twofold. First, each split portion of a biometric template has a smaller dimension compared to its entirety. As such, this enables early rejection of a non-matching biometrics during the authentication phase. Second, no single entity has in possession of the full biometric template of any user except the data owner. The partial biometric templates are encrypted to alleviate the risk of leakage of a user’s biometric information.

- Our protocol is secure against collusion between two or more organizations. Collusion of organizations and cloud provider can never recover the full biometric template of any user without the secret encryption keys that are known only by the trusted entity hosting the national database.
- We implemented PBio as a desktop and an Android applications. Both applications connect to an AWS cloud service, which stores encrypted biometric databases and supports our authentication protocol. We invited 20 volunteers to validate the feasibility and usability of PBio. Our results show that a round-trip authentication process on average takes 74.49ms (desktop) and 626.53ms (Android). These times cover face capture, feature extraction, and execution of all cryptographic operations; out of which our protocol on average takes just 10.06ms and 54.11ms. We also compared our system against recent works [13, 29, 31, 89]. The performance of PBio is comparable, although our overhead is marginally higher at the cryptographic protocol level. This is understandable as our protocol is designed to support cross-organizational authentication, while the prior works consider a direct user-to-organization authentication setting.

2 Overview

PBio consists of four parties: (1) data owner, a fully-trusted entity, who owns the raw biometric templates, shares the encrypted biometric database with a cloud provider and data subscribers, and distributes the user secret keys; (2) cloud provider, a honest-but-curious entity, who receives and stores, from the data owner, a first database of encrypted partial biometric templates, and helps to verify a user without decrypting the encrypted templates; (3) data subscribers, honest-but-curious entities, such as organizations or service providers (e.g. banks), who wish to authenticate their clients. They receive and store, from the data owner, a second database of encrypted partial biometric templates, and uses the database to authenticate a user without decrypting the encrypted templates; and (4) users, (e.g. clients of a bank), whom can be dishonest but must be physically present when register with the national biometric database, submit their biometric in-
Figure 1: PBio Setup and Registration Phase. (1) Data owner generates a master key. The key is used to derive a user secret key for every user. The user secret key is used to derive a subscriber key for every data subscriber, and the subscriber key is also used to derive a verification key; (2) Data owner encrypt-then-split each user’s raw biometric template into two encrypted partial templates using the user secret key. One set of encrypted partial templates is passed to the cloud provider; (3) Data owner uses the subscriber keys to encrypt each user’s encrypted partial template in another set of encrypted partial templates. The resulting set of re-encrypted partial templates, namely subscriber encrypted template, is given to the subscribers; (4) Data owner delivers user secret keys to the users’ registered devices.

formation for authentication using registered tamper-proof devices.

2.1 System Flow

PBio consists of three phases, namely Setup Phase, Registration Phase (Figure 1), and Authentication Phase (Figure 2). Setup Phase. Data owner initializes the system and generates user secret keys to encrypt every user biometric templates. The encrypted templates are split into two parts and distributed to the cloud provider and the data subscriber i respectively. Registration Phase. For every registered data subscriber i, the data owner generates subscriber i encrypted templates and passes it to the subscriber i. For every registered user, the data owner delivers the user secret key to the registered device. Authentication Phase. The users submit the biometrics through a tamper-proof device, which extracts and encrypts using the stored user secret keys. The device then runs the authentication protocol with the data subscriber i and the cloud provider respectively.

2.2 System Goals

PBio’s main goal is to perform cross-organizational biometric authentication without revealing the biometric information:

1. The data subscriber is able to authenticate a user without decryption.
2. The encrypted biometric templates stored by the cloud and data subscribers should leak no information about the users’ biometrics.
3. The freshly submitted biometric features during the authentication phase should remain secret.

2.3 Threat Model

We define the following attackers:

• The attacker that resides in the cloud provider or the data subscribers. We assume the cloud provider and data subscribers to be honest-but-curious. Both follow the protocol honestly but will try to learn the users’ biometric information from the shared encrypted database and during the authentication process.

• The attacker that holds users’ tamper-proof devices. We assume users are authenticated and the keys are distributed by the data owner in the registration phase. We note that the users’ tamper-proof devices are fully trusted, which store the user secret keys and honestly run the proposed protocol. Such devices have been deployed in India’s Aadhaar biometrics authentication [58, 59] using Trusted Execution Environment [20]. Each registered device acts as an oracle to extract and encrypt a user’s biometrics, and proceed with authentication protocol. Hence, the attacker, who holds the victim’s device, may try to manipulate the communication in order to masquerade a victim user and be accepted by the authentication protocol under the victim’s name. An attacker may also collude with cloud provider or subscribers in breaking the stored encrypted biometric templates.
2.4 Definition

PBio consists of the following algorithms and protocols:

- \( msk \leftarrow \text{MKGen}(1^k) \): On input security parameter \( 1^k \), it generates a master secret key \( msk \).
- \( sk_{ID} \leftarrow \text{KGen}(msk, ID) \): On input \( msk \) and the user unique identity \( ID \), it generates a user secret key \( sk_{ID} \).
- \( \vec{c}_{ID} \leftarrow \text{Enc}(sk_{ID}, \vec{x}_{ID}) \): On input \( sk_{ID} \) and a user biometric template as a vector \( \vec{x}_{ID} = (x_1, \cdots, x_n) \), it computes an encrypted vector \( \vec{c}_{ID} \).
- \( \vec{c}_{ID,i}^t, vk_{ID,i} \leftarrow \text{ReEnc}(sk_{ID}, \vec{c}_{ID,i}) \): On input \( sk_{ID}, \vec{c}_{ID} \) and subscriber identity \( i \), it derives a subscriber key and computes a subscriber encrypted vector \( \vec{c}_{ID,i}^t \). A verification key \( vk_{ID,i} \) is also derived from the subscriber key.
- \( t_{ID,i} \leftarrow t\text{Enc}(sk_{ID,i}) \): On input \( sk_{ID} \), subscriber identity \( i \), and a threshold \( t \) that serves to authentication a person, it computes an encrypted threshold \( t_{ID,i} \).
- \( \{d_0, \theta\} \leftarrow \text{SubAuth}(vk_{ID,i}, (\vec{c}_{ID,i}^t, c_{ID,i}^t), t_{ID,i}) \): On input a verification key \( vk_{ID,i} \), a tuple of subscriber encrypted vectors \( (\vec{c}_{ID,i}^t, c_{ID,i}^t) \) and authenticated threshold value \( t_{ID,i} \), it computes their distance \( d_0 \). The output is either \( d_0 \) if the authentication succeeds, or an encrypted authentication result \( \theta \) otherwise.
- \( d_1 \leftarrow \text{CloudAuth}(sk_{ID}, i, (\vec{c}_{ID}, c_{ID,i})) \): On input \( sk_{ID}, i \) and \((\vec{c}_{ID}, c_{ID,i})\), it computes and outputs their distance \( d_1 \).
- \( \theta \leftarrow \text{Combine}(vk_{ID,i}, d_0, d_1, t_{ID,i}) \): On input \( vk_{ID,i}, d_0, d_1, t_{ID,i} \) and \( i \), it combines a full distance \( d = d_0 + d_1 \) and computes an encrypted authentication result \( \theta \).
- \( \{\text{accept, reject}\} \leftarrow \text{Verify}(vk_{ID,i}, \theta) \): On input \( vk_{ID,i} \) and \( \theta \), it decrypts and outputs the authentication result \( \{\text{accept, reject}\} \).

3 Building blocks

PBio requires three main building blocks: (1) a biometric recognition scheme to extract features and construct templates from the raw biometric information (e.g., fingerprint, face, iris); (2) A distance recoverable encryption scheme that we proposed to encrypt these templates; (3) a secure distance computation mechanism that we utilized for authentication.

3.1 Biometric Recognition Scheme

A biometric recognition scheme enables recognition of an individual based on the individual’s biometric (e.g., face) presented to the scheme. We define such a scheme as consisting of the following functions:

- \( \vec{x} \leftarrow \text{Ext(img)} \): On input a raw biometric image \( \text{img} \), it outputs a feature vector \( \vec{x} \in \mathbb{R}^n \).
- \( d \leftarrow \text{Dist}(\vec{x}, \vec{y}) \): On input two feature vectors \( \vec{x} \) and \( \vec{y} \), it outputs a distance \( d \).
- \( \{\text{accept, reject}\} \leftarrow \text{Match}(t, d) \): On input a threshold \( t \) and a distance \( d \), it outputs accept if \( d \leq t^2 \) and reject otherwise.

Given two feature vectors \( \vec{x} = (x_1, \cdots, x_n) \) and \( \vec{y} = (y_1, \cdots, y_n) \), one of the common metrics for matching, which
we use in our protocol, is the squared Euclidean distance $^1$. Hence we define the distance function as: \( \text{Dist}(\vec{x}, \vec{y})^2 = \sum_{i=1}^n (x_i - y_i)^2 = \sum_{i=1}^n x_i^2 + \sum_{i=1}^n y_i^2 - 2 \sum_{i=1}^n x_i y_i \). The authentication result is thus based on the squared Euclidean distance \( d \) in relation to a defined threshold \( t \). In particular, \( x \) and \( y \) belong to the same person if and only if \( \text{Dist}(\vec{x}, \vec{y}) \leq t^2 \). Note that lower value of \( t \) means the system requires higher similarity to pass authentication.

We remark that our protocol deploys the scheme solely to capture facial biometric and output a feature vector. Any biometric recognition scheme based on Euclidean distance can be applied.

### 3.2 Distance Recoverable Encryption

A distance recoverable encryption (DRE) scheme [29, 78, 80, 87, 89] allows one to calculate the distance between two encrypted feature vectors \((\vec{x}, \vec{y})\) is the squared Euclidean distance vectors yet enable computation of the distance between two biometric recognition scheme based on Euclidean distance allows one to calculate the distance between two encrypted feature vectors \((\vec{x}, \vec{y})\). We propose a distance recoverable encryption scheme, DRE, to secure the feature vectors yet enable computation of the distance between two feature vectors in our protocol. The scheme is based on orthogonal matrix, which we define in the followings.

**Orthogonal Matrix.** An orthogonal matrix \( M \) is a \( n \times n \) square matrix such that its inverse and transpose are equal, i.e. \( M^{-1} = M^T \). \( M \) satisfies the following properties:

- Identity transformation: \( M \) and \( M^T \) commute such that \( M^T M = I \), where \( I \) is the identity matrix.
- Product transformation: Given \( M = M_0 M_1 \), if \( M_0 \) and \( M_1 \) are orthogonal matrices, \( M \) is also an orthogonal matrix.
- Preservation of length: Given two pairs of vectors \((\vec{x}, \vec{y})\) and \((\vec{x} M, \vec{y} M)\), their respective Euclidean distances \( \text{Dist} \) are equal as given by \( \text{Dist}(\vec{x}, \vec{y}) = \text{Dist}(\vec{x} M, \vec{y} M) = \|\vec{x} M - \vec{y} M\|^2 \), where \( \|\vec{v}\|^2 = \sum_{i=1}^n v_i^2 \).

The DRE scheme utilizes a pseudorandom function \( \text{PRF}_M(. \rightarrow ) \rightarrow M \in \mathbb{R}^{n \times n} \) that generates pseudorandom orthogonal matrices, a pseudorandom function \( \text{PRF}_V(.) \rightarrow \vec{v} \in \mathbb{R}^n \) that generates pseudorandom vectors, a pseudorandom function \( \text{PRF}_W(.) \rightarrow w \in [1, 0, 2.0] \) that generates a scale factor, and a pseudorandom permutation \( \text{PRP}(., \vec{s}) \rightarrow \pi(\vec{s}) \). The encryption function \( E \) is defined as follows:

- \( \vec{c} = E(sk, \vec{x}) \): Given as input a secret key \( sk \) and a feature vector \( \vec{x} = (x_1, \ldots, x_n) \), the algorithm generates a pseudorandom orthogonal matrix \( m \) from \( \text{PRF}_m(sk) \), a pseudorandom vector \( \vec{v} \) from \( \text{PRF}_V(sk) \) and a scale factor \( w \) from \( \text{PRF}_W(sk) \). It then runs \( \text{PRP}(sk, \vec{x}) \) to permute \( \vec{x} \) resulting in \( \pi(\vec{x}) \). An encrypted vector is generated as \( \vec{c} = w(\pi(\vec{x}) + \vec{v}) M \).

### 3.3 Secure Distance Computation

Bringer et al. proposed a secure distance computation protocol called GSHADE [7]. It allows two parties, a sender \( S \) and a verifier \( V \), to securely compute the distance of two biometric features. It guarantees one party does not get more information about the other party’s inputs than what can be deduced from its own inputs and outputs. A central building block of GSHADE is oblivious transfer (OT). OT is an interactive protocol whereby the sender has a number of messages, and the receiver wishes to obtain a specific message, without the sender knowing which it is, while also ensuring that the receiver gets no information about the other messages which the sender holds. In brief, let \( \vec{x} = (x_1, \ldots, x_k) \) with \( x_i = (x_{1(i-1)1} + \cdots, x_{1(i-1)k}) \) and \( \vec{y} = (y_1, \ldots, y_k) \) with \( y_i = (y_{k(i-1)1} + \cdots, y_{k(i-1)k}) \) are \( n = k \times \ell \)-bit integer vectors. Three functions are defined where \( f_{\lambda}(\vec{x}) = \sum_{i=1}^k x_i^2 \), \( f_{\lambda}(\vec{y}) = \sum_{i=1}^k y_i^2 \), and \( f_{\lambda}(\vec{x}) + f_{\lambda}(\vec{y}) = -2\lambda - x_{k(i-1)1} \cdots x_{k(i-1)\ell} \) for \( i = 1, \ldots, k \) and \( j = 1, \ldots, \ell \). Both \( S \), on input \( \vec{y} \), and \( V \), on input \( \vec{x} \), run the protocol as follows:

1. \( S \) chooses \( n \) random values \( r_1, \ldots, r_n \in \mathbb{Z}_m^* \) and \( S \) and \( V \) run \( OT^{\log_2(m)} \) where:
   - \( V \)’s selection bit is \( x_i \).
   - \( S \)’s input is \( (r_i + f_{\lambda}(\vec{y}), r_i + f_{\lambda}(\vec{y})) \).
   - The output obtained by \( V \) is \( t_i = r_i + f_{\lambda}(x_i, \vec{y}) \).
2. \( V \) computes and outputs \( T = \sum_{i=1}^n t_i + f_{\lambda}(\vec{x}) \).
3. \( S \) computes and outputs \( R = \sum_{i=1}^n r_i - f_{\lambda}(\vec{y}) \).
4. At the end, either \( S \) or \( V \) learns the distance by computing \( \text{Dist}(\vec{x}, \vec{y}) = T - R = d \).

**Theorem 1 (7)** Security is proven by simulation in the OT-hybrid setting, where OTs are simulated by a trusted oracle. We recall that each simulator is provided with the input and output of the corrupted party. Case 1 - \( V \) is corrupted. Since \( V \) receives no messages beyond those in OT, its view can be perfectly simulated. Case 2 - \( S \) is corrupted. Given \( S \)’s output and input \( x \), \( S \)’s view can be perfectly simulated by sending random values \( t_1, \ldots, t_{n-1} \in \mathbb{Z}_m \) and \( t_n = T - \sum_{i=1}^{n-1} t_i - f_{\lambda}(\vec{x}) \) to \( S \) in the OTs.
4 PBio

PBio consists of several components: First, we encrypt biometric templates using DRE. Second, we employ GSHADE to allow two parties with two private vector inputs $\tilde{x}$ and $\tilde{y}$, respectively, to securely decide if the Euclidean distance between $\tilde{x}$ and $\tilde{y}$ is smaller than a given threshold without leaking extra information. Third, we perform periodical update to refresh the encrypted partial biometric templates that are accessible by organizations. We discuss several ways to go about performing such update in Section 7.

4.1 Construction

Let $BR = (\text{Ext}, \text{Dist}, \text{Match})$ to be any biometric recognition scheme based on Euclidean distance, DRE be the distance recoverable encryption scheme, and GSHADE the secure distance computation protocol. We define a keyed-hash message authentication code function $\text{HMAC}()$ and a symmetric encryption function for encryption $\text{AES}_E(\cdot)$ and decryption $\text{AES}_D(\cdot)$.

The proposed scheme consists of a tuple ($\text{Setup}, \text{MKGen}, \text{KGen}, \text{Enc}, \text{ReEnc}, \text{tEnc}, \text{SubAuth}, \text{CloudAuth}, \text{Combine}, \text{Verify}$). We assume the input message is a feature vector $\tilde{x}$, extracted from a user ID biometric image, e.g., $\tilde{x}_{ID}\leftarrow \text{BR.Ext}(\text{img}_{ID})$.

- $\text{msk}\leftarrow \text{MKGen}(1^k)$: The algorithm generates a random string as the master key $\text{msk}\in \mathbb{Z}_p^*$.
- $\text{sk}_{ID}\leftarrow \text{KGen}(\text{msk}, ID)$: The algorithm runs $\text{HMAC}(\text{msk}.ID,\text{time})$ with the input of $\text{msk}$, user unique identity $ID$, and a timestamp $\text{time}$ to generate a user secret key $\text{sk}_{ID}$.
- $\tilde{c}_{ID, i}\leftarrow \text{Enc}(\text{sk}_{ID}, \tilde{x}_{ID})$: The algorithm generates a encrypted vector by running $\tilde{c}_{ID, i}\leftarrow \text{DRE}E.(\text{sk}_{ID}, \tilde{x}_{ID})$.
- $\tilde{c}_{ID,i, j}\leftarrow \text{ReEnc}(\text{sk}_{ID}, \tilde{c}_{ID,i, j}, i)$: The algorithm runs $\text{HMAC}(\text{sk}_{ID}, ID, i, \text{time})$ to generate a new subscriber $i$ key $\text{sk}_{ID,i}$ and $\text{HMAC}(\text{sk}_{ID,i})$ to generate a new verification key $\text{vk}_{ID,i}$. The rest is similar to Enc, which generates a subscriber encrypted vector by running $\tilde{c}_{ID,i, j}\leftarrow \text{DRE}E.(\text{sk}_{ID,i}, \tilde{x}_{ID,i, j})$.
- $\tilde{t}_{ID,i} = \text{tEnc}(\text{sk}_{ID}, i, t)$: The algorithm runs $\text{PRF}_\text{v}(\text{sk}_{ID}) \rightarrow w_{ID}$ and $\text{PRF}_\text{y}(\text{sk}_{ID}, i) \rightarrow w_{ID,i}$. It then encrypts a threshold $\tilde{t}_{ID,i} = w_{ID,i}.t$.
- $\{d_0, \theta\} \leftarrow \text{SubAuth}(\text{vk}_{ID,i}, (\tilde{c}_{ID,i, j, ID}, \tilde{c}_{ID,i, j, ID}), (\tilde{t}_{ID,i}))$: An interactive protocol that runs GSHADE between a device and a subscriber $i$ where the device input is a tuple $(\tilde{c}_{ID,i, j, ID}, \tilde{t}_{ID,i})$ and subscriber $i$ input is $\tilde{c}_{ID,i, j}$. At the end of the protocol, the device receives a distance $d_0 = \text{Dist}(\tilde{c}_{ID,i, j, ID}, \tilde{c}_{ID,i, j, ID})$ and runs $b \leftarrow \text{BR.Match}(\tilde{t}_{ID,i}, d_0)$. The device returns encrypted authentication result $\theta \leftarrow \text{AES}_E(b, \tilde{c}_{ID,i, j, ID})$, which indicates an early rejection to the authentication. Otherwise, the user proceeds to CloudAuth.

- $d_1 \leftarrow \text{CloudAuth}(\text{sk}_{ID, i}, (\tilde{c}_{ID,i, j, ID}))$: An interactive protocol that runs GSHADE between a device and a cloud provider where the device input is a tuple $(\text{sk}_{ID, i}, \tilde{c}_{ID,i, j, ID})$ and the cloud input is $\tilde{c}_{ID,i, j, ID}$. At the end of the protocol, the device receives and computes a distance $d_1 = \text{Dist}(\tilde{c}_{ID,i, j, ID}, \tilde{c}_{ID,i, j, ID}) \cdot \text{PRF}_\text{v}(\text{sk}_{ID, i})$. The cloud receives nothing and the device receives $d_1$.
- $\theta = \text{Combine}(\text{vk}_{ID,i}, 0, d_0, d_1, \text{t}_{ID,i})$: On input $\text{vk}_{ID,i}, d_0, d_1, \text{t}_{ID,i}$ and $i$, the algorithm generates a full distance $d = d_0 + d_1$ and runs $b \leftarrow \text{BR.Match}(\text{t}_{ID,i}, d)$. The output is the encrypted authentication result $\theta \leftarrow \text{AES}_E(b, \text{vk}_{ID,i})$.
- $\{\text{accept}, \text{reject}\} \leftarrow \text{Verify}(\text{vk}_{ID,i}, \theta)$: On input $\text{vk}_{ID,i}$ and $\theta$, it decrypts and outputs the authentication result $\{\text{accept}, \text{reject}\} \leftarrow \text{AES}_D(\text{vk}_{ID,i}, \theta)$.

**Correctness.** Given the elements $(\tilde{c}_{ID,i, j, ID}, \tilde{c}_{ID,i, j, ID}, \tilde{c}_{ID,i, j, ID}, \tilde{c}_{ID,i, j, ID})$ generated by the defined algorithms above, the following equations always hold. If $\text{BR.Dist}(\tilde{x}_{ID,i, j}, \tilde{y}_{ID}) \leq t^2$, then $d_0 \leftarrow \text{SubAuth}((\tilde{c}_{ID,i, j, ID}), \text{ID}, i) \leftarrow \text{CloudAuth}(\text{sk}_{ID,i}, (\tilde{c}_{ID,i, j, ID}))$, Else if $\text{BR.Dist}(\tilde{x}_{ID,i, j}, \tilde{y}_{ID}) > t^2$ and early rejection occurs, then $\theta \leftarrow \text{Combine}(\text{vk}_{ID,i}, d_0, d_1, \text{t}_{ID,i})$, and accept $\leftarrow \text{Verify}(\text{vk}_{ID,i}, \theta)$; Else if $\text{BR.Dist}(\tilde{x}_{ID,i, j}, \tilde{y}_{ID}) > t^2$, then $d_0 \leftarrow \text{SubAuth}(\text{vk}_{ID,i}, (\tilde{c}_{ID,i, j, ID}, \tilde{c}_{ID,i, j, ID}), (\tilde{t}_{ID,i}))$. If $\theta = \text{Combine}(d_0, d_1, \text{t}_{ID,i})$, and reject $\leftarrow \text{Verify}(\text{vk}_{ID,i}, \theta)$; Else if $\text{BR.Dist}(\tilde{x}_{ID,i, j}, \tilde{y}_{ID}) > t^2$, then $\theta \leftarrow \text{Combine}(\text{vk}_{ID,i}, (\tilde{c}_{ID,i, j, ID}, \tilde{c}_{ID,i, j, ID}), \text{ID}, i) \leftarrow \text{CloudAuth}(\text{sk}_{ID,i}, (\tilde{c}_{ID,i, j, ID}, \tilde{c}_{ID,i, j, ID}), (\text{ID}, i))$, $d_1 \leftarrow \text{tEnc}(\text{sk}_{ID}, i, \text{t}_{ID,i})$, and reject $\leftarrow \text{Verify}(\text{vk}_{ID,i}, \theta)$.

4.2 PBio System

We now present the three phases of PBio, utilizing the construction previously described.

**Setup Phase.** The data owner possesses a set of biometric templates, which is pre-collected from all the users. Each record of a user contains the user’s identity information $ID$ and the biometric image $\text{img}$. The data owner generates a master key and then derives the user secret key for every user. The data owner encrypts each template into two parts. The encrypt-then-split approach allows the data owner to provide a partial copy of the encrypted templates to the cloud provider, and the other partial copy to be prepared for the data subscriber $i$. The details of the setup protocol is described in Figure 3.

**Registration Phase.** The registration phase enables a subscriber or a user to register to the data owner respectively. In other words, a new subscriber registers and receives the encrypted templates, and a user registers a device to install the user secret key. Thus it consists of two sub-phases: (a) subscriber registration and (b) user registration. The details of the registration protocol is described in Figure 4. We also elaborate how the key distribution works in Section 7.3.
The setup protocol

Input:
- Data Owner: User’s raw biometric image, user’s identity information, and proof of identity.

Output:
- Data owner: Encrypted database $\langle c_0|c_i \rangle = \hat{c}_i$ and the user table $Table_i$.
- Cloud provider: Encrypted partial database $\hat{c}_i$.

Protocol:
1. The data owner executes the master key generation algorithm to generate a master secret key $msk \leftarrow \text{MKGen}(1^k)$.
2. The data owner then runs a biometric recognition scheme $BR$ to extract the biometric feature and stores the biometric template $\bar{x} \leftarrow BR.\text{Ext}(img)$.
3. For every user with unique identity $ID$, the data owner runs the key generation algorithm to generate user secret key $skID \leftarrow \text{MKGen}(msk,ID)$. The data owner stores $(skID, ID, time)$ in a user table $Table_i$.
4. For every user biometric template $\bar{x} = \{\bar{x}_1, \cdots, \bar{x}_m\}$ where $m$ is the total number of users, data owner generates the encrypted database by running $\hat{c}_i \leftarrow \text{Enc}(skID, \bar{x}_i)$.
5. Data owner splits by half the encrypted database into two parts $\hat{c}_0|\hat{c}_i = \hat{c}_i$. The first part $\hat{c}_0$ will be applied during the registration phase and the second part $\hat{c}_i$ is outsourced to a cloud provider.

Figure 3: Setup phase

The registration protocol

Input:
- Data owner: Encrypted partial database $\hat{c}_0$ and the user table $Table_i$.
- Subscriber $i$: Unique identity $i$ along with the proof of identity.
- User: Unique identity $ID$ along with the proof of identity.

Output:
- Subscriber $i$: Subscriber encrypted templates $\hat{c}_{ID,i}$ and verification key $vkID,i$.
- User: User secret key $skID$.

Protocol:
1. The protocol is initialized in a secure manner by (a) the subscriber or (b) the user:
   a. The subscriber $i$ submits the proof of identity and registers to the data owner to request for a copy of subscriber encrypted templates.
   b. User submits the proof of identity with the registered device and requests for a user secret key $skID$.
2. Data owner performs the following steps:
   a. Verifies the provided registration information and generates the subscriber encrypted template and verification key pair $(\hat{c}_{ID,i}, vkID,i)$ by running $\text{ReEnc}(skID, \hat{c}_{ID,i})$ using every user secret key and encrypted partial template $(skID, \hat{c}_{0,0})$.
   b. Verifies the user identity and embeds $skID$ to the user registered device.

Figure 4: Registration phase

5 Security of PBio

5.1 Security Models

We follow the security formulation in [17] for an authentication scheme, which includes correctness, soundness, and zero-knowledge. We remark that the correctness definition for biometric authentication is slightly different from transitional definition (as in [17]), since a legitimate user might be rejected with a small probability—that’s the definition of false rejection rate or false negative rate, due to the noise in measurements of biometric feature. In the real world scenarios, the user may re-try after some adjustment (e.g. adjust face angle).

In this work, every secret key used to encrypt the biometric templates for every user is derived from a master secret key owned by the data owner. The encrypted biometric templates are stored by the cloud provider and the data subscribers respectively. We allow collusion between the cloud and the data subscribers. The goal of the adversary is to masquerade a victim user and be accepted by the authentication solution.
The authentication protocol

**Input:**
- User: Freshly captured biometric image $img_{ID}$ along with his or her ID and secret key $sk_{ID}$.
- Subscriber $i$: Encrypted template and verification key pair $(c_{i,ID}, vk_{ID,i})$ that belongs to the user ID.
- Cloud: Encrypted partial template $c'_{i,ID}$ that belongs to user ID.

**Output:**
- Both the user and subscriber $i$ obtain the authentication result.

**Protocol:**
1. The user scans his biometric image $img_{ID}$ with a tamper-proof device. The device then runs $\tilde{y}_{ID} \leftarrow \mathsf{BR}.\mathsf{Ext}(img_{ID})$ to extract the feature vector and runs $c_{i,ID} \leftarrow \mathsf{Enc}(sk_{ID}, \tilde{y}_{ID})$ to generate an encrypted biometric feature vector. Additionally, the device splits by half the encrypted biometric features into two parts, where $(c'_{i,0ID} | c'_{i,1ID}) = c_{i,ID}$. The device re-encrypts $c_{i,0ID}$ by running $\tilde{c}_{i,1ID}, vk_{ID,i} \leftarrow \mathsf{ReEnc}(sk_{ID}, c'_{i,0ID}).$
2. The device then runs $(d_0, \theta) \leftarrow \mathsf{SubAuth}(vk_{ID,i}, (\tilde{c}_{i,1ID}, c_{i,1ID}), t_{ID,i})$ with subscriber $i$ to compute the first partial distance, where the device has input $(vk_{ID,i}, c_{i,1ID}, t_{ID,i})$ and subscriber $i$ has input $c_{i,1ID}$. The device verifies the first part of the authentication process and proceeds if and only if $d_0 \leq t_{ID,i}$ where $t_{ID,i} \leftarrow t\mathsf{Enc}(sk_{ID}, i)$. Otherwise, the process stops and the device returns the encrypted authentication result as $\theta \leftarrow \mathsf{AES}(vk_{ID,i}, \mathsf{reject})$ to subscriber $i$, which indicates an early rejection.
3. The second partial distance $d_1 \leftarrow \mathsf{CloudAuth}(sk_{ID}, i, (c_{i,1ID}, c_{i,1ID}))$ is run with the cloud provider where the device has input $c_{i,1ID}$ and the cloud has input $c_{i,1ID}$.
4. The device combines and checks $(d_0, d_1)$ to obtain the encrypted authentication result $\theta \leftarrow \mathsf{Combine}(d_0, d_1, t_{ID,i})$. The device returns $\theta$ to subscriber $i$.
5. Finally, subscriber $i$ decrypts and obtains the authentication result $\langle \mathsf{accept, reject} \rangle \leftarrow \mathsf{Verify}(vk_{ID,i}, \theta)$.

Figure 5: Authentication phase

5.2 Security Analysis

The proposed protocol utilized by our PBio system applies the distance recoverable encryption (DRE) scheme in Section 3.2 and secure distance computation protocol (GSHADE) in Section 3.3. Hence, its security depends on the security of these underlying schemes.

**Security of DRE** We note that the inherent limitation of our DRE is that, it suffers from linear attack. Precisely, the adversary can find the secret key by solving a large linear equation system, if this adversary obtains sufficient pairs of plaintexts and ciphertexts [48]. As shown in [80], such DRE scheme is only secure against ciphertext-only attack. To overcome the limitation of DRE in our system, we ensure that every user uses different encryption key $sk$. Basically, we use DRE like “One-Time Pad” from the view of potential adversaries. In legitimate usage, the encryption key will never be re-used for different objects. We remark that, the official authentication client app in our solution is a trusted party, it holds the per-user secret key, and can access multiple copies of ciphertexts under that secret key.

**Theorem 2 (Security of our DRE)** Let $\tilde{x}$ and $\tilde{y}$ denote two points in the plaintext domain, and $\tilde{c}$ is any valid ciphertext generated using our DRE where the encryption key is randomly chosen from its domain. We have $\Pr[\tilde{x} | \tilde{c}] = \Pr[\tilde{y} | \tilde{c}]$, which means a single ciphertext leaks no information of the plaintext. Let real number $t \in (0, 1)$ be the threshold. Given a ciphertext, there are at least $1/t^n$ number of possible plaintexts under distinct encryption keys, such that the distance between every two plaintexts is at least $2t$.

The proof of Theorem 2 is given in Appendix A.1.

**Proposition 2 (Correctness)** Our proposed protocol is correct, i.e. any legitimate user who follows the authentication protocol exactly by submitting his or her biometric image, will be authenticated successfully, except for a small probability (i.e. the false negative rate of biometric feature).

The above proposition follows directly from the property of DRE and correctness of GSHADE.

**Theorem 3 (Zero-Knowledge)** After interacting with a user by executing our protocol for many times, both the cloud provider and the subscriber $i$ learn nothing about the user’s biometric raw data, beyond the ciphertext.

The proof of Theorem 3 is given in Appendix A.2.

**Theorem 4 (Soundness)** Probabilistic polynomial time adversary (even colluded with some subscribers and the cloud provider), cannot pass our authentication with non-negligible probability.

The proof of Theorem 4 is given in Appendix A.3.
6 Implementation and Evaluation

This section provides the efficiency assessment of our construction (Section 4.1 and 4.2). We also compare our construction with existing works to demonstrate practicality of our proposal.

6.1 Evaluation of the Proposed Construction

The experiment focuses on the performance overhead introduced by Enc, ReEnc, SubAuth, CloudAuth, Combine, and Verify, as they are the core components. Our implementation uses a python face recognition library\(^2\) as the biometrics recognition scheme. We remark that any biometric recognition scheme can be utilized as long as it is based on Euclidean distance. We also use NumPy library\(^3\) to generate vectors and matrices and perform the matrix operations. We conducted the experiment on a desktop with Intel Core i7-8700 CPU @3.20GHz with 8GB RAM and two cores.

Similar to [13, 29, 89], a set of random vectors was generated to represent the original biometric template database because one can apply any biometric recognition scheme to extract the feature vectors in practice. We randomly generated \(m \times n\) vectors. This means there are \(m\) number of user in the database with \(n\) dimension of biometric feature vector. For the remainder of this section, we denote 64-\(n\), 128-\(n\), 320-\(n\), and 640-\(n\) to represent dimensions of 64, 128, 320 and 640 respectively. The experimental results in Table 1 shows the time required by Enc for \(m \times n\) biometric template database.

Enc performs the first layer encryption only. We require to re-encrypt again by running ReEnc algorithm for every subscriber for half of the dimension as the second layer encryption. In PBio, we apply face recognition scheme that consists of 128-\(n\) for a template. We then split 128-\(n\) into half after the first layer encryption and we re-encrypt the second layer encryption in 64-\(n\). The first layer encryption time took approximate 1.14ms per user and an additional 0.61ms is required for the second layer encryption, which indicates that PBio encryption requires 1.74ms in total. We summarized the encryption time per user in Table 2. We notice that the encryption time increases with the dimensional size of a template.

We list the sizes of biometric template databases used in our experiment in Table 3. We note that the size of a biometric template relies on the size of dimension \(n\). In our case the size of the original database and the encrypted database are the same because our encryption technique transforms an original value into a random value of the same size. For our experiment, the size of the biometric template in 128-\(n\) is 1024 bytes (B), its encrypted template in 128-\(n\) is also 1024B.

We then analyze the performance of PBio authentication protocol in Table 4. This experiment is categorized into two parts, namely Part I and Part II. In Part I, the freshly submitted biometric features with 128-\(n\) was first encrypted into a subscriber encrypted features, which run Enc and ReEnc on inputs 128-\(n\) and 64-\(n\) respectively, and then we applied SubAuth for the first partial authentication. We then run CloudAuth and Combine as the second partial authentication steps in Part II. If output of SubAuth is reject, then early rejection occurs and Part II will not be executed. Finally, we run Verify to receive the authentication result. Overall, PBio took approximate 2.776ms + 2\(\ell\) in total for the authentication where \(\ell\) is the network latency for GSHADE. We noticed that \(\ell\) is very much affected by the network itself. In our environment, we first tested in our local machine, which achieved the result in Table 4.

The goal of the partial authentication is to achieve early rejection. For example, in the scenario where in Part I authentication is rejected, the process can be terminated without proceeding to Part II. This reduce cost of the communication and network latency \(\ell\). Later, we connected the machines over the internet for our second experiment in Section 6.3. The estimated value for \(\ell\) is around 3.5ms. The communication overhead is only around 2.83KB for a full authentication and 1.42KB in the case of early rejection.

\(\text{\footnotesize Table 1: Computation Overhead of Enc algorithm}\)

<table>
<thead>
<tr>
<th>No. of User</th>
<th>64-(n)</th>
<th>128-(n)</th>
<th>320-(n)</th>
<th>640-(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.58ms</td>
<td>1.14ms</td>
<td>4.32ms</td>
<td>20.79ms</td>
</tr>
<tr>
<td>1,000</td>
<td>0.6s</td>
<td>1.17s</td>
<td>4.51s</td>
<td>20.74s</td>
</tr>
<tr>
<td>10,000</td>
<td>5.93s</td>
<td>11.73s</td>
<td>44.82s</td>
<td>5m29s</td>
</tr>
<tr>
<td>100,000</td>
<td>58.83s</td>
<td>1m58s</td>
<td>7m35s</td>
<td>34m45s</td>
</tr>
<tr>
<td>1,000,000</td>
<td>9m49s</td>
<td>19m36s</td>
<td>1h15m</td>
<td>5h47m</td>
</tr>
<tr>
<td>2,000,000</td>
<td>19m25s</td>
<td>39m8s</td>
<td>2h31m</td>
<td>11h33m</td>
</tr>
<tr>
<td>5,000,000</td>
<td>49m7s</td>
<td>1h35m</td>
<td>6h1m</td>
<td>28h52m</td>
</tr>
<tr>
<td>10,000,000</td>
<td>1h37m</td>
<td>3h4m</td>
<td>12h30m</td>
<td>57h44m</td>
</tr>
</tbody>
</table>

\(\text{\footnotesize Table 2: PBio Encryption Performance Per User in PBio}\)

<table>
<thead>
<tr>
<th>PBio Encryption Time</th>
<th>64-(n)</th>
<th>128-(n)</th>
<th>320-(n)</th>
<th>640-(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enc Time</td>
<td>0.55ms</td>
<td>1.14ms</td>
<td>4.52ms</td>
<td>20.79ms</td>
</tr>
<tr>
<td>ReEnc Time</td>
<td>0.24ms</td>
<td>0.60ms</td>
<td>1.43ms</td>
<td>4.53ms</td>
</tr>
<tr>
<td>Total Encryption Time</td>
<td>0.82ms</td>
<td>1.74ms</td>
<td>5.95ms</td>
<td>25.32ms</td>
</tr>
</tbody>
</table>

\(\text{\footnotesize Table 3: Size of Biometric Templates}\)

<table>
<thead>
<tr>
<th>No. of User</th>
<th>Size of Template Database</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>64-(n)</td>
</tr>
<tr>
<td>1</td>
<td>512B</td>
</tr>
<tr>
<td>1,000</td>
<td>512KB</td>
</tr>
<tr>
<td>10,000</td>
<td>5.12MB</td>
</tr>
<tr>
<td>100,000</td>
<td>51.2MB</td>
</tr>
<tr>
<td>1,000,000</td>
<td>512GB</td>
</tr>
<tr>
<td>2,000,000</td>
<td>1.024GB</td>
</tr>
<tr>
<td>5,000,000</td>
<td>2.56GB</td>
</tr>
<tr>
<td>10,000,000</td>
<td>5.12GB</td>
</tr>
</tbody>
</table>

\(^2\)https://pypi.org/project/face-recognition/
\(^3\)https://pypi.org/project/numpy/
6.2 Accuracy

We also tested the accuracy of our construction. A LFW dataset [30] with the provided test case was extracted, which consists of 409 true positive pairs and 444 true negative pairs. We follow the default threshold \( t = 0.6 \). The matched result is 402 out of 409, which shows 98.29\% of them is true positive, and non-matched result is 442 out of 444, which shows 99.55\% is true negative. The proposed construction achieved the same accuracy in both the true positive and true negative results, while the early rejection is 190 out of 442, which is 42.99\% in the non-matched results.

6.3 Performance Evaluation of PBio System

We simulated the application environment of PBio system through an Android application and AWS cloud. Readers may refer back to Fig 1 and Fig 2 for the PBio system architecture. The experiment was conducted in a real usage setting, where we recruited 20 volunteers, collected their face images as the experiment dataset, and for them to test the Android application and connecting to the AWS cloud using their mobile devices. We subscribed to Amazon Elastic Compute Cloud (EC2) to serve as the cloud provider and a subscriber. Both of the EC2 machines are running in Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz, single core and 1GB RAM. The source code of PBio System can be found in [4].

During the setup phase, a 256-bits master secret key \( msk \) was randomly selected. We then created a biometric template database from each of the volunteer’s face image. Again, we use the same face recognition scheme that consists of 128-\( n \) dimension for a template. The template database was then encrypted following the proposed PBio system. We generated a set of encrypted database \( \vec{c}_i = \{ \vec{c}_i^{ID} | \vec{c}_i^{ID} \} \), and \( \vec{c}_i^{ID} \) was stored by the cloud provider. We used \( \vec{c}_i^{m} \) to generate subscriber \( i \) encrypted template \( \vec{c}_i^{ID} \) and passed it to subscriber \( i \).

Fig. 6 shows the screenshots of PBio Android application used in the experiment. Fig. 6(a) shows a screenshot when user setups the device to retrieve the user secret key. Fig. 6(b) shows a list of services offered by the PBio’s subscriber. Fig. 5(c) shows a screenshot when the person succeeded in authentication. Fig. 6(d) shows a screenshot when a person failed in authentication.

As in our use case where individuals first register their biometric information coupled with their identity proof with an organization, we assume that user will always provide an identity information that would allow us to verify before the user secret key \( sk_{ID} \) is forwarded to the respective user device. We note that the volunteers were using their own Android devices. During the authentication protocol, a device captured a user face image \( img_{ID} \) and run the face recognition scheme to generate the feature vector \( \vec{y}_{ID} \). To verify with subscriber \( i \), the device runs the proposed encryption scheme to generate both encrypted vector \( \vec{c}_i^{ID} \) and re-encrypted vector \( \vec{c}_i^{SID} \) respectively. The device then verifies with subscriber \( i \) and proceeds to verify with the cloud if and only if the first part authentication succeeded. The final authentication result returns \( accept \) if the similarity is within the threshold. We asked the volunteers to repeat the authentication process for more than 10 times to find the average.

Table 5 shows the summary of execution time for both the desktop and the Android platform setting. The implementation and experiment of the desktop version was discussed in Section 6.1. It represents a kiosk based client device. The experiment result shows the total authentication time required by executing the biometric recognition scheme (i.e. Extraction) and the authentication protocol. The experiment was conducted in a practical daily usage scenario, where the volunteers perform face authentication using their mobile devices.

In summary, a round-trip authentication process takes 74.49ms and 626.53ms in desktop platform and Android platform respectively, which includes the total transmission time. The communication overhead can be found in Table 4. We note that any biometric recognition scheme can be deployed as long as it is based on Euclidean distance, which means that one can replace it with a more efficient scheme.
6.4 Comparison with Existing Methods

In this section, we compare the execution performance of PBio with works in [13, 29, 31, 89]. We compare the encryption and verification performance for 128-n. We note that [13] was implemented using the Paillier encryption scheme [60] with 1024-bit modulus, and [29, 89] use the similar technique as ours. For the most recent work in [31], we compare our performance with the experiment available in [31], which was implemented in Android platform using the linear homomorphic encryption (LHE) scheme [35]. As shown in Table 6, we noticed that our encryption scheme is slower by a factor of $2.4 \times$ compared to [89] and the verification time needed is slower by a factor of $5.58 \times$ and $13.49 \times$ compared to [29] and [89] respectively. This is due to [29] and [89] enable one to find the verification result without secure multiparty computation (SMC).

We would like to emphasize that these existing works focus on the direct user-to-organization setting, and do not provide cross-organizational authentication as in PBio. However, our goal is similar such that to prevent leaking the stored, encrypted templates to be decryptable by any party, and protect the input biometric features from leaking. As shown previously, our distance-preserving encryption also allows one to find the distance without decryption, but we found the insecurity, so we applied GSHADE to overcome the weakness. Besides, it is obvious that [13] and [31], who applied homomorphic encryption (HE), require higher computation overheads. Table 7 shows the comparison of the encrypted database size. We notice PBio ciphertext size is smaller than all the previous works as we neither extend vector dimensions nor introduce additional elements.

7 Discussion

We may adopt a number of additional measures in our biometric system to further enhance its security. Firstly upon registration, the data owner assigns a unique key to each user. This ensures that the resulting encryption applied to each raw biometric template will be distinct for different users. Secondly, our biometric system is enabled to refresh the partial encrypted database held by the data subscriber and the cloud provider either periodically or when is it necessary. For instance, in the event a user’s device is lost and requires a replacement. For the remainder of this section, we provide a comparison between two feasible mechanisms for template encryption as well as a detailed discussion on the key update process. We also elaborate the key distribution in practice and leave it as an open problem.

7.1 encrypt-then-split vs split-then-encrypt

We further consider two different approaches to perform authentication: encrypt-then-split and split-then-encrypt as shown in Table 8. The encrypt-then-split approach first encrypts the raw biometric template and split them into two. For the latter approach, the raw biometric template is first split and each individual split portion is subsequently encrypted. Recall that PBio first checks the Part I and proceeds with the authentication: encrypt-then-split as shown in Table 8. The encrypt-then-split approach first encrypts the raw biometric template and split them into two. For the latter approach, the raw biometric template is first split and each individual split portion is subsequently encrypted. Recall that PBio first checks the Part I and proceeds with the rest if and only if Part I is successfully passed. Since split-then-encrypt approach performs encryption in half of the n-dimension, we see that split-then-encrypt approach achieves faster early rejection as the encryption time needed for Part I and II can be done separately. However, the split-then-encrypt approach reduces half of the n-dimension, which may weaken the security guarantee. In general, a higher dimension corresponds to a larger security parameter. As such, we decide to adopt the encrypt-then-split mechanism in our experiments of PBio and leave this security analysis as future work.
### Table 8: Comparison of Authentication Performance under encrypt-then-split and split-then-encrypt.

<table>
<thead>
<tr>
<th>Authentication Time</th>
<th>encrypt-then-split</th>
<th>split-then-Encrypt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part I</td>
<td>2.21ms + ( t )</td>
<td>1.05ms + ( t )</td>
</tr>
<tr>
<td>Part II</td>
<td>0.48ms + ( t )</td>
<td>1.06ms + ( t )</td>
</tr>
<tr>
<td>Total Time</td>
<td>2.69ms + 2( t )</td>
<td>2.11ms + 2( t )</td>
</tr>
</tbody>
</table>

\( t \) denotes the network latency

### 7.2 Key Update

Periodic key updates of existing database help to safeguard against potential keys leakage or exposure. In addition, should a group of users’ keys be compromised, a timely key update process ensure that their biometric templates are still protected. We discuss several methods which enables efficient key updates.

**Method 1:** When the key update for a user with identity information \( ID \) is initialized, the device receives \( \tilde{c} \) \(_{ID} \) and \( \tilde{c}' \) \(_{ID} \) from the cloud provider and the relevant data subscribers respectively. The device updates the encrypted template of the user by performing \( \text{Enc}(sk_{\text{new}}, \tilde{c}_{ID}) \) and \( \text{ReEnc}(sk_{\text{new}}, \tilde{c}'_{ID}) \) which are subsequently transmitted to the cloud and corresponding data subscribers respectively. Consequently, \( \text{Enc}(sk_{\text{new}}, \tilde{c}'_{ID}) \) is the updated encrypted template held by the cloud while \( \text{ReEnc}(sk_{\text{new}}, \tilde{c}'_{ID}) \) is the updated encrypted template of held by the data subscribers. A potential limitation of this method is that the device is required to fetch encrypted templates of the associated users from the data subscribers and cloud provider whenever a key update process is called upon.

**Method 2:** One other feasible way is for the data owner to be involved in the key update process. In this way, whenever a key update process is called upon for a group of users, the data owner can simply generate new keys and send the corresponding new encrypted templates of these users to the cloud and data subscribers. The device will also be notified of the generation and values of these new keys. However, this requires the data owner to be online during every key update process.

**Method 3:** To overcome the above issues and limitations, we introduce a trusted key management server to be involved in the key update process. This key server which can be continuously online is hosted by the data owner. The main role of this key server is to issue new keys whenever a key update process is initiated. When the key update for the user is initialized, the key server fetches \( \tilde{c}_{ID} \) and \( \tilde{c}'_{ID} \) from the cloud and data subscribers respectively. The key server decrypts these encrypted templates to obtain \( \tilde{x}_{ID} \). New keys are generated to perform a re-encryption of \( \tilde{x}_{ID} \) as similar to the setup and registration protocols. The new keys are sent to the trusted device while the newly updated encrypted templates are sent to the cloud and data subscribers.

### 7.3 Key Distribution

PBio system adopted the key distribution technique following India’s Aadhaar technical report [58]. Once a user provides a sufficient proof of identity via PBio Android application, the device generates a public and private key pair using Android built-in trustzone. The data owner enrolls the device public key and encrypts the user key before returning to the device. For now, the device stores the ciphertext, which can only be decryptable by its private key and used for PBio authentication phase within the trustzone. Since PBio offers solution to secure biometrics templates and authenticate a person without decryption. We, therefore, leave the secure key distribution as a future work.

### 8 Related Work

**Insecurity of Biometric Templates.** There are numerous methods enabling the reconstruction of biometric images from certain types of raw biometric templates. In general, the security considerations relating to biometric templates can be classified into two: (1) template inversion \([9, 10, 23, 25, 28, 47, 49, 62, 65, 73]\), which adversary tries to reconstruct a biometric template into a raw image; (2) hill climbing \([2, 24, 50–52, 76, 83]\), which adversary tries to construct synthetic biometric templates that can pass the authentication. This highlights that storing a biometric template in the clear is not sufficient to protect its underlying biometric information.

**Techniques to Secure Biometric Templates.** There are two generic solutions to protect biometrics \([27]\), namely image processing \([42, 57, 72]\) and cryptographic techniques \([14, 27, 63, 66, 81]\). Image-based techniques are computationally efficient but as stated in \([82]\), most of the image processing techniques result in decreasing accuracy due to the distortion applied on the original image. Biohashing \([8, 38]\) was proposed to transform an individual biometric feature into a unique value that is used to perform exact match during authentication. Cancellable biometrics \([3, 33, 64]\) applies similar techniques like biohashing, but it allows revocation if the stored value is compromised. There are, however, successful attacks on cancellable biometrics and biohashing \([12, 41, 44, 46]\). Fuzzy commitment and secure sketch ensure the privacy of biometrics by providing information-theoretic guarantees using error correcting codes \([19, 36, 37, 71, 84]\) or signal embeddings \([15, 54, 55, 70, 77]\). Chun et al. \([14]\) proposed a fuzzy extraction technique to create a metadata during enrollment, and a secret token is recovered from the metadata to complete the authentication. However, these techniques face security issues when it is used multiple times and it assumes certain conditions on the distribution of biometrics, as stated in \([37]\). Chatterjee et al. \([11]\) proposed a scheme that protect biometric templates based on secure sketches. It prevents an attacker from learning the owner of biometric information.
templates by collecting and randomly permuting multiple fingerprints of the users.

Cryptographic techniques commonly known as secure computation, on the other hand, preserve accuracy comparable to that of the ordinary recognition schemes but incur higher performance overhead. Secure computation can be achieved by applying various cryptography tools such as Homomorphic Encryption (HE) [53, 60, 69], Predicate Encryption (PE) [39, 68, 89], Inner Product Encryption (IPE) [1, 18, 40], Oblivious Transfer (OT) [32, 56], and Garbled Circuit Evaluation (GCE) [85]. These tools allow the verification to be done securely via calculating the distance between the enrolled and probed biometric templates in the encrypted domain [5, 6, 79], but it is also known that most of the cryptography-based techniques are computation intensive. Barni et al. [5] applied Paillier’s HE [60] which allows user to verify whether the submitted biometric feature is in the server database.

Privacy-Preserving Biometric Authentication. Numbers of privacy-preserving biometric authentication (and identification) systems were proposed which can be categorized into two settings. The first is a direct user-to-organization setting as can be seen from the system proposed by Šeděnka et al. [67]. The system allows every user, with a trusted device that holds a secret key, to enroll his encrypted biometric templates using homomorphic encryption (HE). During authentication, garbled circuit evaluation (GCE) is applied to decrypt and compare the similarity among the encrypted template and the submitted feature. It represents a common setting whereby the organization (data owner) hosts the (encrypted) biometric templates. Similar to [67], the biometrics systems by Zhou and Ren [89], PassBio, and Lee et al. [45] also require a trusted device that encrypts the biometric features during enrollment and authentication by using predictable encryption (PE) and inner production encryption (IPE) respectively. The merit of [45, 89] is that decryption is not required during authentication as PE and IPE allows one to find the similarity given two encrypted templates.

The second is an outsourced setting where the data owner securely outsource the authentication (and identification) processes to a cloud [34, 43] in order to reduce the computation and operational costs. Chun et al. [13] proposed a system that uses HE to encrypt the biometric templates. The encrypted templates and secret key are distributed to two clouds respectively with the assumption that no collusion happens. During authentication, the fresh encrypted biometric features are submitted to the two clouds to find the similarity using HE and GCE so that the cloud can neither learn the plain biometric templates nor the features. Xiang et al. [81] proposed a scheme which relies on a hybrid encryption scheme, which is a variant of fully homomorphic encryption scheme. However, it results in higher communication overheads as compared to the previous schemes. Tian et al. [75] introduced remote user authentication that is similar to [45, 67, 89] where user biometric templates are stored in the encrypted format and the authentication is done in an anonymous and unlinkable manner with a cloud. A matrix based technique was proposed by Yuan and Yu [87] that allows the cloud to verify an individual without decryption. However, [87] was found to be insecure and enhanced by [78], CloudBI. Further enhancement was proposed in [29], which also stated that such the matrix based technique is weakly secure. Guo et al. [27] applied randomization technique which allows feature extraction and authentication to be done in the encrypted domain. However, it supports only face recognition and the security is not guaranteed as only completeness analysis was provided. Recently, Im et al. [31] applied linear HE to construct a scheme under the outsourced setting, yet it requires around 1 second for authentication.

9 Conclusions

We proposed a new privacy-preserving biometric authentication system, PBio, which allows an owner of a raw biometric database to share a derived database to other organizations to authenticate users in a privacy-preserving manner. By doing so, the organizations provide authentication services without needing to collect biometrics or invest in securing raw biometric databases. Furthermore, a user needs to register only once, that is, with the data owner to access services provided by different organizations. The proposed system supports biometric recognition schemes that is based on euclidean distance. The accuracy is preserved even in the encrypted domain. Besides, we introduced an encrypt-then-split construction, which allows the data owner to split the encrypted biometric templates into two or more copies. One copy is then given to a cloud provider and the other copies to other subscribed organizations. With this property, the proposed system allows user to authenticate a partial encrypted biometric feature with the partial encrypted biometric template stored by the organization, which computes a partial result that allows the organization to make early rejection. In the case if the final result is necessary, the cloud provider is required to be involved in ascertaining the final result. Assuming data breach happens in either the cloud or the organization, the attacker will only obtain partial information of the templates. We developed a prototype for the proposed system. The experiment shows that our proposed system is practical.

References


[2] Andy Adler. Sample images can be independently restored from face recognition templates. In CCECE


[74] Irene Tham. Facial recognition will remove need for passwords, ids, 2020.


A Proof sketch

A.1 Proof of Theorem 2

Part I. Recall that our DRE is

\[ \text{Enc}(sk, \vec{x}) = w \cdot (\vec{x} + \vec{v}) \cdot M \]  

where \( sk = (w, \vec{v}, M) \). Let encryption key \( sk = (w, \vec{v}, M) \) and ciphertext \( \vec{c} = \text{Enc}(sk, \vec{x}) = w \cdot (\vec{x} + \vec{v}) \cdot M \), we have

\[ \vec{v} = f(w, M, c, \vec{x}) = w^{-1} \cdot \vec{c} \cdot M^{-1} - \vec{x} \]

Therefore, we show that the conditional probability \( \text{Pr}[\vec{x} | \vec{c}] \) is independent on \( \vec{x} \):

\[
\text{Pr}[\vec{x} | \vec{c}] = \text{Pr}[sk = (w, \vec{v}, M) \text{ where } \vec{v} = f(w, M, c, \vec{x})]
\]

\[
= \sum_{w,M} \text{Pr}[sk = (w, \vec{v} = f(w, M, c, \vec{x}), M)]
\]

\[
= \frac{\#w \times \#M}{\#sk},
\]

where \( \#w \) denotes total number of possible scaling factor, \( \#M \) denotes total number of possible orthogonal matrix of dimension as indicated in our DRE scheme, and \( \#sk \) denotes the total number of possible encryption keys in our scheme. As a result, for any two distinct plaintexts \( \vec{x} \) and \( \vec{y} \), we have

\[ \text{Pr}[\vec{x} | \vec{c}] = \text{Pr}[\vec{y} | \vec{c}] \]

Part II. Let \( t \) be the threshold. We request the distance between the two points \( \vec{x} \) and \( \vec{y} \) to be larger than \( 2t \), so they cannot represent the same bio-object. Then we count how many such distinct points with pairwise distance \( \geq 2t \). Within the \( n \)-dimension cube \([-1.0, 1.0]^n\), each person’s biometric measurement could be treated as a \( n \)-dimension sphere with center \( u \) and radius \( t \), where \( u \) is the measurement of the dimension during the registration phases. So the total number \( N \) of such \( n \)-dimension sphere is

\[ N \geq \frac{2.0^n}{(2r)^n} = (1/t)^n. \]

For example, in our experiment, \( t = 0.6 \).

A.2 Proof of Theorem 3

Our authentication protocol is a nested application of two schemes whose keys are chosen independently: (1) Our distance recoverable encryption DRE; (2) The GSHADE secure two party computation scheme. Recall that the GSHADE scheme allows two parties with private vector inputs \( \vec{x} \) and \( \vec{y} \) respectively, to compute the Euclidean distance between \( \vec{x} \) and \( \vec{y} \), without leaking extra partial information. In the application of GSHADE in our proposed scheme, the private input (\( \vec{x} \) or \( \vec{y} \)) is the ciphertext of a raw biometric feature vector, generated from our DRE. At the end of GSHADE, the user (the authentication client software) will obtain the distance between \( \vec{x} \) and \( \vec{y} \) as output, and the Cloud (or Subscriber) will obtain a single bit (i.e. acceptance or rejection) as output. Due to the zero-knowledge feature of GSHADE scheme (Theorem 1), the execution of GSHADE as a subroutine of our protocol, will not leak any more information about private inputs \( \vec{x}, \vec{y} \) beyond the corresponding outputs. Since we assume the user is running in a trusted environment, the user checks the threshold and returns the authentication result 1 or 0, which is a single bit: if \( \text{Dist} (\vec{x}, \vec{y}) \leq t^2 \) holds, to the other party (Cloud or Subscriber).

According to our Theorem 2, the ciphertext \( \vec{y} \) possess by the subscriber \( i \) or cloud provider (who may collude with an adversary) leaks no information about the plaintext, which is the underlying raw biometric feature vector. Therefore, our nested application of GSAHDE and DRE ensures that no extra information leaks during the protocol, except the final computation result, which is a single bit.

A.3 Proof of Theorem 4

First of all, we remark that, the authentication client software in user’s device is trusted, and is verified by the authentication server every time, before user starts to authenticate to the server. Thus, our official authentication client software is the only way to authenticate to the server, and third party authentication client software can be easily detected and rejected by authentication server.

The adversary may collude with both the cloud provider and subscriber \( i \), and thus is able to find the DRE ciphertext \( \text{ct} \) of user’s bio template vector \( \vec{x} \), and observe any network communications of GSHADE. Note that in our invocation of GSHADE protocol between authentication client software and cloud (or subscriber), the authentication server learns only one bit information—accepting or rejecting this user. Furthermore, due to Theorem 2, a single ciphertext does not leak any information of plaintext, i.e. the user’s bio template vector \( \vec{x} \). Consequently, the adversary is unable to find an estimation \( \vec{x}' \) such that \( \text{Dist}(\vec{x}, \vec{x}') \) is smaller than the given threshold, and thus cannot pass our authentication scheme.