Practical Privacy-Preserving Face Authentication for Smartphones Secure Against Malicious Clients

Jong-Hyuk Im, Seong-Yun Jeon, and Mun-Kyu Lee, *Member, IEEE*

***Abstract*— We propose a privacy-preserving face authentication system for smartphones that guarantees security against malicious clients. Using the proposed system, a face feature vector is stored on a remote server in encrypted form. To guarantee security against an honest-but-curious server who may try to learn the private feature vector, we perform a Euclidean distance-based matching score computation on encrypted feature vectors using homomorphic encryption. To provide security against malicious clients, we adopt a blinding technique. We implement the proposed system on a mobile client and a desktop server. Through an experiment with real-world participants, we demonstrate that secure face verification can be completed in real time (within 1.3 s) even when a smartphone is involved, with an Equal Error Rate (EER) of 3.04%. In further experiments with two public face datasets, CFP and ORL, face verification is completed in approximately 1 s with EER of 1.17% and 0.37%, respectively. Our system is two orders of magnitude faster than previous privacy-preserving face verification method with the same security assumptions and functionalities. To achieve this secure real-time computation, we improve the Catalano-Fiore transformation which converts a linear homomorphic encryption scheme into a quadratic scheme, and parallelize the decryption procedure of our system.**

***Index Terms*— Privacy-preserving authentication, biometric verification, residual network (ResNet), homomorphic encryption, secure two-party computation.**

I. Introduction

BIOMETRIC authentication is a method of performing authentication using a user's biometric data such as the face, fingerprints, iris, and palmprint. There are two types of biometric authentication [1]: biometric identification and biometric verification. First, *biometric identification* is a one-to-many matching which identifies the owner of the input biometric data from a database storing multiple users' data. The output of this task is the identifier (ID) of the matched user if any. By contrast, in *biometric verification*, the biometric data of a person who wants to be authenticated as a certain ID is compared with the stored data associated with that ID. This is a one-to-one comparison, whose output is the similarity score (or biometric distance) between the input data and the stored data, and the decision whether to accept this person (input). With the recent advances in mobile technologies, many smart devices are adopting biometric verification as a means to authenticate their owners. It is used not only for unlocking the device, but also for authorizing the user to execute security-critical applications such as financial services.

For biometric verification, the user's biometric data should be stored first during the user's registration, using a data structure called a biometric template. In many cases, these templates are represented as feature vectors. However, biometric data are unique and unchangeable, which implies that the leakage of these data may cause more critical privacy issues than the compromise of traditional passwords and personal identification numbers (PINs). Moreover, we frequently witness cases where a user's biometric template is compromised, in particular for mobile devices [2]. Therefore, it is necessary to develop a biometric authentication method that keeps the biometric templates secure.

In the literature, there are two well-known approaches for protecting biometric templates. The first one converts biometric data using noninvertible transforms such as cancelable biometrics [3], [4] and fuzzy commitment [5]. However, this approach carries the disadvantage in that the biometric accuracy may be affected owing to the transform. Moreover, there are many cases where a one-way transform is analyzed and successfully inverted [6]–[8]. The other approach stores the biometric template on a remote server outside the device [9], [10]. The server works as a secure storage. This approach also fits the situation where a user wants to be authorized for access to a certain remote service using an authentication server. However, this approach may cause another privacy issue in that the remote server may learn the user's biometric data and maliciously use them. This issue raises the need to design privacy-preserving authentication protocols [10]–[12].

There are various tools for constructing privacy-preserving authentication protocols, including homomorphic encryption (HE) [13]–[20], Yao's garbled circuit (GC) [21] and SPDZ protocols [22], [23]. For these protocols, we must consider two types of adversaries [24], [25]: an *honest-but-curious (HBC) adversary* and a *malicious adversary*. An HBC adversary only tries to learn the private data of the other party while normally following the protocol. Meanwhile, a malicious adversary manipulates the protocol to learn the private data of the other party or to generate false output. It is more realistic to consider a malicious adversary, in particular for a client, which reflects the situation where a smartphone is stolen or compromised.

In this paper, we propose a privacy-preserving face verification system that can be performed in slightly more than 1 second on a smartphone with negligible communication overhead.

Manuscript received March 13, 2019; revised August 1, 2019 and December 16, 2019; accepted January 9, 2020. Date of publication January 27, 2020; date of current version February 6, 2020. This work was supported in part by Basic Science Research Program through the National Research Foundation (NRF) funded by Ministry of Education (MOE), South Korea, under Grant 2017R1D1A1A09000915 and in part by Korea Electric Power Corporation under Grant R18XA01. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Julien Bringer. *(Corresponding author: Mun-Kyu Lee.)*

The authors are with the Department of Computer Engineering, Inha University, Incheon 22212, South Korea (e-mail: imjhyuk@gmail.com; roland.korea@gmail.com; mklee@inha.ac.kr).

Digital Object Identifier 10.1109/TIFS.2020.2969513

The system implements a two-party computation (2PC) protocol using homomorphic encryption as well as state-of-the-art face processing techniques. To the best of our knowledge, this is the first work on a practical two-party mobile biometric authentication which enables the real-time computation of a Euclidean distance-based matching function, and maintains security against malicious clients.

A. Related Works

In 2009, Erkin *et al.* proposed the first privacy-preserving biometric identification protocol [26], where the client owns a face image and the server owns a database containing many users' face images. After running the protocol, the client obtains the identifier of the matched user if any, but it does not learn any useful information about the content of the server's database. Neither the client's face image nor the identification result is revealed to the server. This protocol was an HE-based 2PC protocol, where the server and client may be HBC adversaries. Its efficiency was improved in [27] by combining HE and GC. In 2010, Osadchy *et al.* proposed a privacy-preserving face identification system called SCiFI [28] in which a new index-based feature extraction technique was used instead of the legacy *eigenfaces* method [29]. In 2011, privacy-preserving fingerprint identification protocols [30], [31] and a protocol for both fingerprint and iris identification [32] were also published. In 2015, Demmler *et al.* proposed ABY (for Arithmetic, Boolean, and Yao sharing), a new secure 2PC framework that efficiently combines various secret sharing schemes, and showed how this framework can be applied to privacy-preserving biometric identification [33]. All of the above protocols [26]–[28], [30]–[33] share the same security goal, and consider an HBC client and an HBC server. All users' original biometric data are stored, in unencrypted form, into the server's database.

Note that the security assumption of the above identification protocols is completely different from that of biometric verification. For privacy-preserving biometric verification involving a remote server, we require that the client's biometric template is encrypted and stored into the server. Recently, various studies on verification protocols considering these conditions were conducted [34]–[43]. These protocols can be classified into protocols that consider only HBC parties [34]–[37], and the protocols that consider malicious parties [38]–[43].

In 2014, Chun *et al.* proposed a protocol for matching the fingerprint of a claimed user with an encrypted fingerprint template stored on a remote server using HE and GC for an HBC client and server [34]. In 2016, Cheon *et al.* proposed a protocol to support Hamming distance computations required for iris recognition using HE and message authentication codes (MACs) [35]. In the same year, Im *et al.* [36] proposed a protocol to support the Euclidean distance computation using HE and a transformation function [44] that converts a linear HE into a quadratic HE. In 2017, Lin *et al.* proposed privacy-preserving face verification involving four HBC parties [37].

It is reasonable to assume that even a dishonest server will behave in a semi-honest manner, i.e., it is HBC, because it wants to keep its reputation as a service provider and is interested in computing a correct result, e.g., a score [26]. However, as for clients, it is more realistic to assume that they may behave in an arbitrary malicious way. To meet this requirement, Šeděnka *et al.* designed a privacy-preserving face verification protocol using GC in 2015 [39]. This protocol can calculate either the Euclidean or Manhattan distance while ensuring the security against a malicious client, but has shown somewhat impractical performance. In the same year, Shahandashti *et al.* proposed a privacy-preserving implicit authentication protocol using absolute average deviation (AAD) [38]. Their protocol achieved security against malicious clients using HE and a cut-and-choose technique. Gasti *et al.* [40] and Abidin [41] also proposed biometric verification protocols secure against malicious adversaries, but these involve a third party helper for secure computation between the client and server. In 2018, Gunasinghe and Bertino proposed a privacy-preserving face verification protocol using a support vector machine (SVM) and *eigenfaces* [42]. Their protocol performs a zero-knowledge proof of knowledge (ZKPK) with the help of a device equipped with trusted execution environment (TEE). The protocol involves a client, a server, and a trusted third party, but interestingly, only the client and trusted third party participate in a registration phase, and only the client and server participate in an authentication phase. In the same year, Droandi *et al.* [43] proposed a protocol that allows very fast biometric matching using the SPDZ protocol [22].

B. Our Contributions

In this paper, we propose a practical two-party privacy-preserving biometric verification system. To be precise, our contributions are as follows:

- We propose a real-time biometric verification system for smartphones that supports efficient computation of the squared Euclidean distance (SED), which is one of the most universal matching functions for biometrics, while ensuring security against malicious clients and HBC servers. A client and server can run the proposed protocol without a third party.
- We implement the proposed method with various state-of-the-art face image processing techniques for smartphones. According to the results of our experiment involving real-world users, a face verification can be completed within approximately 1.3 second with an EER of 3.04%.
- For faster computation, we improve the Catalano-Fiore transformation [44], [45], which converts a linear homomorphic encryption scheme into a quadratic scheme. Using the improved transformation, the computational complexity of decrypting a quadratic ciphertext is halved. Although this result was directly used to accelerate the most time-consuming part in the proposed system, the improved transformation is of independent interest. We also parallelize the decryption procedure in our system using OpenMP [46].

II. Preliminaries

A. Biometric Authentication

Among the two types of biometric authentication, we concentrate on biometric verification. Fig. 1 shows a generic procedure for face verification [47]. Other biometric verification systems use a similar process. Specifically, the face verification operates as follows: (1) An image including a face is input using a sensor such as a camera. (2) A region of

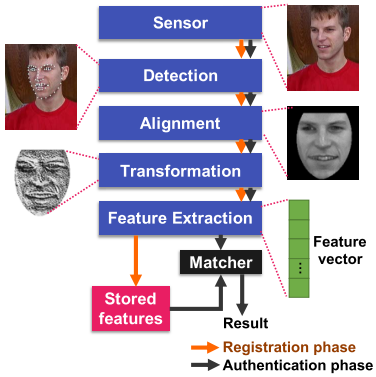

Fig. 1. Generic procedure for face verification.

the face is detected from the input image, and landmarks are extracted from the detected face [48]. (3) The image is cropped according to the region of the face and the cropped face is aligned (frontalized) using the extracted landmarks [49]. (4) When required, transformations such as filters or local binary patterns (LBPs) are performed to normalize the aligned face image [50]. (5) A feature vector is extracted from the aligned and transformed image using extraction methods such as Principal Component Analysis (PCA) [29] and deep neural networks [47], [51]. In this paper, we use the well-known residual network (*ResNet*) proposed by He *et al.* [51] as a deep neural network for image recognition. (6) In the registration phase, the extracted feature vector is stored in a database. In the authentication phase, a matching score, e.g., the Euclidean distance, between the stored feature vector and a feature vector to be authenticated is calculated, and then an authentication result is determined. If the feature vector is stored in an encrypted form to preserve privacy, a biometric matcher must compute the score on the ciphertext space. However, typical matching functions are not represented linearly. For example, because the squared Euclidean distance $\mathsf{SED}(\boldsymbol{v}, \boldsymbol{u}) = \Sigma_{i=1}^{k}(v_i - u_i)^2$ between two vectors $\boldsymbol{v} = (v_1, \ldots, v_k)$ and $\boldsymbol{u} = (u_1, \ldots, u_k)$ is a quadratic operation, the encryption scheme should support quadratic operations on the ciphertext space.

B. Linear Homomorphic Encryption (LHE)

Homomorphic encryption (HE) is an encryption scheme that supports operations on a ciphertext space. Let Enc and Dec be the encryption and decryption operations, respectively. Let $\boxplus$ be an operation on the ciphertext space. We say this encryption scheme is a linear homomorphic encryption (LHE) scheme, if $\mathsf{Dec}(\mathsf{Enc}(m_1) \boxplus \mathsf{Enc}(m_2)) = m_1 + m_2$ for any two plaintexts m_1 and m_2. Since the concept of homomorphic encryption was first proposed by Rivest *et al.* in 1978 [13], several LHE schemes have been proposed [14], [15], [20]. An LHE scheme $\widehat{\mathcal{HE}}$ can be represented by four probabilistic polynomial time (PPT) algorithms: $(\widehat{\mathsf{KeyGen}}, \widehat{\mathsf{Enc}}, \widehat{\mathsf{Eval}}, \widehat{\mathsf{Dec}})$.

- $\widehat{\mathsf{KeyGen}}(1^\lambda)$ generates a private key sk and a public key pk according to a given security parameter λ. The generated public key pk contains information about the plaintext space $\mathcal{M}$.
- $\widehat{\mathsf{Enc}}(m, pk)$ encrypts a plaintext $m \in \mathcal{M}$ to a ciphertext $C \in \widehat{\mathcal{C}}$ using the public key pk, where $\widehat{\mathcal{C}}$ denotes a ciphertext space and $\mathcal{M}$ denotes a plaintext space.
- $\widehat{\mathsf{Dec}}(C, sk)$ outputs a decrypted plaintext m when a ciphertext C and a private key sk are given.
- $\widehat{\mathsf{Eval}}(f, C_1, \ldots, C_t, pk)$ outputs an evaluated ciphertext, given a linear arithmetic circuit $f : \mathcal{M}^t \to \mathcal{M}$, pk and ciphertexts $C_1, \ldots, C_t$. Let $\boxplus$, $\boxminus$ and $\cdot$ denote the homomorphic addition, subtraction and constant multiplication, respectively, s.t. $\widehat{\mathsf{Dec}}(\widehat{\mathsf{Enc}}(a, pk) \boxplus \widehat{\mathsf{Enc}}(b, pk), sk) = a + b$, $\widehat{\mathsf{Dec}}(\widehat{\mathsf{Enc}}(a, pk) \boxminus \widehat{\mathsf{Enc}}(b, pk), sk) = a - b$ and $\widehat{\mathsf{Dec}}(a \cdot \widehat{\mathsf{Enc}}(b, pk), sk) = a \times b$ for any two plaintexts a and b. Then, $\widehat{\mathsf{Eval}}$ can be expressed as $a_1 \cdot C_1 \odot \cdots \odot a_t \cdot C_t$ using t constants $a_i \in \mathcal{M}$, where $\odot$ represents either $\boxplus$ or $\boxminus$. For convenience, we write $a_1 \cdot C_1 \boxplus \cdots \boxplus a_t \cdot C_t$ as $\boxplus_{i=1}^{t} a_i \cdot C_i$.

C. Transforming LHE to Evaluate Degree-2 Equation

We need an HE that supports quadratic operations, i.e., multiplications on ciphertexts, to calculate the Euclidean distance on the encrypted data. Fully homomorphic encryption (FHE) scheme, supporting an arbitrary number of additions and multiplications on ciphertexts, e.g. [17]–[19] as well as their restricted version, somewhat homomorphic encryption, could be the ultimate solution. However, they do not yet guarantee a sufficient speed for real-time applications such as the one considered in this paper. Prior to the development of FHE, Boneh *et al.* have proposed an HE scheme that supports quadratic operations on encrypted data [16], but this scheme was still slow. In 2015, Catalano and Fiore proposed a transformation function that transforms an LHE scheme, e.g., [15], [20], into an HE scheme that supports quadratic operations on encrypted data [44]. An HE scheme constructed using this transformation was very fast compared to [16].

Now, we explain the Catalano-Fiore transformation [44]. According to [45], which is the full version of [44], a semantically secure quadratic homomorphic encryption scheme $\mathcal{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ can be constructed using a semantically secure public-space LHE scheme $\widehat{\mathcal{HE}}$ as follows:

- $\mathsf{KeyGen}(1^\lambda)$ performs $\widehat{\mathsf{KeyGen}}(1^\lambda)$ of $\widehat{\mathcal{HE}}$ according to a given security parameter λ to obtain a key pair (pk, sk). It is assumed that pk contains the plaintext space $\mathcal{M}$ and the ciphertext space $\widehat{\mathcal{C}}$ of $\widehat{\mathcal{HE}}$, and the plaintext space of $\mathcal{HE}$ is equal to $\mathcal{M}$.
- $\mathsf{Enc}(m, pk)$ computes $C \leftarrow (a, \beta) = (m - b, \widehat{\mathsf{Enc}}(b, pk)) \in \mathcal{M} \times \widehat{\mathcal{C}}$, using an input plaintext $m \in \mathcal{M}$ and an arbitrary plaintext $b \in_R \mathcal{M}$, where $\in_R$ denotes selecting uniformly an arbitrary element from a set.
- $\mathsf{Eval}(f, C_1, \ldots, C_t, pk)$ is composed of five different procedures: $(\mathsf{Add}_1, \mathsf{Mult}, \mathsf{Add}_2, \mathsf{cMult}_1, \mathsf{cMult}_2)$. Add_1 and Mult work only with level-1 ciphertexts, i.e., the immediate results of Enc or their linear combinations. On the other hand, Add_2 adds two level-2 ciphertexts, i.e., the two results of Mult. In addition, cMult is a procedure that multiplies a ciphertext with a constant. The procedure, like Add, can branch to cMult_1 and cMult_2 depending on the level of the input ciphertexts.
  - Add_1: Given two level-1 ciphertexts $C_1, C_2 \in \mathcal{M} \times \widehat{\mathcal{C}}$, where $C_i = (a_i, \beta_i)$ for $i = 1, 2$, produces a level-1 ciphertext $C = (a, \beta) \in \mathcal{M} \times \widehat{\mathcal{C}}$ as follows:

$$a \leftarrow a_1 + a_2, \quad \beta \leftarrow \beta_1 \boxplus \beta_2.$$

– Mult: Given two level-1 ciphertexts $C_1, C_2 \in \mathcal{M} \times \widehat{\mathcal{C}}$, where $C_i = (a_i, \beta_i)$ for $i = 1, 2$, produces a level-2 ciphertext $C = (\alpha, \boldsymbol{\beta}) \in \widehat{\mathcal{C}} \times \widehat{\mathcal{C}}^2$ as follows:

$$\alpha \leftarrow \widehat{\mathsf{Enc}}(a_1 \times a_2, pk) \boxplus a_1 \cdot \beta_2 \boxplus a_2 \cdot \beta_1,$$
$$\boldsymbol{\beta} \leftarrow (\beta_1, \beta_2)^\top.$$

– Add_2: Given two level-2 ciphertexts C_1, C_2, where $C_i = (\alpha_i, \boldsymbol{\beta}_i) \in \widehat{\mathcal{C}} \times \widehat{\mathcal{C}}^{2\times l_i}$ for $i = 1, 2$, produces a level-2 ciphertext $C = (\alpha, \boldsymbol{\beta}) \in \widehat{\mathcal{C}} \times \widehat{\mathcal{C}}^{2\times(l_1+l_2)}$ as follows:

$$\alpha \leftarrow \alpha_1 \boxplus \alpha_2, \quad \boldsymbol{\beta} \leftarrow [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2].$$

– cMult_1: Given a constant $c \in \mathcal{M}$ and a level-1 ciphertext $C = (a, \beta) \in \mathcal{M} \times \widehat{\mathcal{C}}$, produces a level-1 ciphertext C' as follows:

$$C' \leftarrow (c \times a, c \cdot \beta) \in \mathcal{M} \times \widehat{\mathcal{C}}.$$

– cMult_2: Given a constant $c \in \mathcal{M}$ and a level-2 ciphertext $C = (\alpha, \boldsymbol{\beta}) \in \widehat{\mathcal{C}} \times \widehat{\mathcal{C}}^{2\times l}$, where $\boldsymbol{\beta} = [(\beta_{1,1}, \beta_{2,1})^\top, \ldots, (\beta_{1,l}, \beta_{2,l})^\top]$, this produces a level-2 ciphertext $C' = (\alpha', \boldsymbol{\beta}') \in \widehat{\mathcal{C}} \times \widehat{\mathcal{C}}^{2\times l}$ as follows:

$$\alpha' \leftarrow c \cdot \alpha, \quad \boldsymbol{\beta}' \leftarrow [(c \cdot \beta_{1,1}, \beta_{2,1})^\top, \ldots, (c \cdot \beta_{1,l}, \beta_{2,l})^\top].$$

- $\mathsf{Dec}(C, sk)$: When a ciphetext C is given, one of the following decryption procedures are performed according to the level of the ciphertext.
  – A level-1 decryption decrypts a level-1 ciphertext $C = (a, \beta) \in \mathcal{M} \times \widehat{\mathcal{C}}$ using the private key sk as follows:

$$m \leftarrow a + \widehat{\mathsf{Dec}}(\beta, sk).$$

  – A level-2 decryption decrypts a level-2 ciphertext $C = (\alpha, \boldsymbol{\beta}) \in \widehat{\mathcal{C}} \times \widehat{\mathcal{C}}^{2\times l}$, where $\boldsymbol{\beta} = [(\beta_{1,1}, \beta_{2,1})^\top, \ldots, (\beta_{1,l}, \beta_{2,l})^\top]$, using the private key sk as follows:

$$m \leftarrow \widehat{\mathsf{Dec}}(\alpha, sk) + \Sigma_{i=1}^{l} \{\widehat{\mathsf{Dec}}(\beta_{1,i}, sk) \times \widehat{\mathsf{Dec}}(\beta_{2,i}, sk)\}. \quad (1)$$

III. System Model

We consider a system involving a user, the user's smartphone device and an authentication server. The user is a party that wants to receive a service from the authentication server using a smartphone device. The authentication server verifies the identity of the claimed user by executing a face verification protocol with the device and provides the service only to the registered user.

It is assumed that the user's device does not store any information except its user's identifier ID, public parameters and a key pair. That is, it does not store the feature vector of its user. It is assumed that the server does not store any information except for the public parameters and the encrypted feature vectors, associated to ID, obtained during the registration phase.

When we describe the proposed verification protocol, we will use the term "client" to refer to the device that is controlled by both its human user's action and the software installed in it, following the convention in the literature [42].

Based on the above system model, we consider the following adversaries.

- **Honest-But-Curious Server** (**HBCS**) is a server-side adversary that aims to obtain the face image of the client's user, its feature vector, and/or the private key, but performs the protocol honestly and correctly with the client.
- **Malicious Client** is a client-side malicious adversary. We consider a situation in which a malicious user gains control of the legitimate user's device, e.g., by stealing or compromising it. Throughout the paper, a malicious client means this compromised device is controlled by the malicious user with malicious software. Therefore, the stored private key of the client is available to the adversary, but the legitimate user's face image is not directly available because it is not stored locally in the device. There are two types of malicious clients: **MC1** aims to obtain feature vectors stored in the server by maliciously manipulating the protocol. **MC2** aims to be successfully authenticated by maliciously manipulating the protocol without any information about the face image or the feature vector of the legitimate user.

IV. Proposed Methods

In this section we propose a method to calculate SED efficiently, improving on the transformation in [45], and propose a privacy-preserving face verification system applying the improved transformation to an LHE scheme, while implementing state-of-the-art face processing techniques.

A. Improved Transformation for Squared Euclidean Distance

The performance of a transformed HE $\mathcal{HE}$ [45] depends on the performance of the underlying LHE $\widehat{\mathcal{HE}}$. In particular, according to (1), the decryption cost for a level-2 ciphertext of $\mathcal{HE}$ is $(2l + 1)\times$(cost of $\widehat{\mathsf{Dec}}$), where l is the length of the ciphertext. When l is large, this may take considerable time. However, this cost can be significantly reduced if the quadratic equations to be evaluated are restricted to the sum of squares instead of the sum of general degree-2 terms. This modified functionality is useful for the efficient computation of SED, which is required for computing a matching score.

In this section, we propose a modified transformation technique that transforms an LHE into a semantically secure quadratic homomorphic encryption scheme $\mathcal{HE}'$, where the evaluation of quadratic expressions is restricted to sums of squares. To be precise, the transformed HE is defined as $\mathcal{HE}' = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$, where KeyGen, Enc, Eval, and Dec are defined as the same as in Section II-C, except that Mult in Eval is replaced by Square and the level-2 decryption is slightly modified. First, Square is defined as follows:

Square: Given a level-1 ciphertext $C = (a, \beta) \in \mathcal{M} \times \widehat{\mathcal{C}}$, this produces a level-2 ciphertext $C' = (\alpha', \boldsymbol{\beta}') \in \widehat{\mathcal{C}} \times \widehat{\mathcal{C}}^2$ as follows:

$$(\alpha', \boldsymbol{\beta}') \leftarrow (\widehat{\mathsf{Enc}}(a^2, pk) \boxplus (2a \cdot \beta), (\beta, 0)^\top).$$

Next, the decryption of a level-2 ciphertext $C' = (\alpha', \boldsymbol{\beta}') \in \widehat{\mathcal{C}} \times \widehat{\mathcal{C}}^{2\times l}$ is redefined as follows:

$$m \leftarrow \widehat{\mathsf{Dec}}(\alpha', sk) + \sum_{i=1}^{l} \{\widehat{\mathsf{Dec}}(\beta'_{1,i}, sk)\}^2, \tag{2}$$

where $\boldsymbol{\beta}' = [(\beta'_{1,1}, 0)^\top, \ldots, (\beta'_{1,l}, 0)^\top]$. As a result, the decryption of a level-2 ciphertext is completed by calling $\widehat{\mathsf{Dec}}$ only $l+1$ times instead of $2l+1$ times. The following theorem shows that $\mathcal{HE}'$ is semantically secure. The proof is given in the appendix.

Theorem 1: If $\mathcal{HE}$ is semantically secure, then $\mathcal{HE}'$ is semantically secure.

Now, we explain how $\mathcal{HE}'$ accelerates the computation of SED. In the original scheme $\mathcal{HE}$, the encrypted value $S = (\alpha, \boldsymbol{\beta})$ of SED between two vectors $(m_1, \ldots, m_k)$ and $(m'_1, \ldots, m'_k)$ is calculated from the encrypted vectors $\boldsymbol{C} = ((a_1, \beta_1), \ldots, (a_k, \beta_k)) = (\mathsf{Enc}(m_1, pk), \ldots, \mathsf{Enc}(m_k, pk))$ and $\boldsymbol{C}' = ((a'_1, \beta'_1), \ldots, (a'_k, \beta'_k)) = (\mathsf{Enc}(m'_1, pk), \ldots, \mathsf{Enc}(m'_k, pk))$ as follows:

$$\alpha \leftarrow \boxplus_{i=1}^{k}\{\widehat{\mathsf{Enc}}(\bar{a}_i^2, pk)\} \boxplus \{\boxplus_{i=1}^{k}(\bar{a}_i \cdot \bar{\beta}_i \boxplus \bar{a}_i \cdot \bar{\beta}_i)\},$$
$$\boldsymbol{\beta} \leftarrow [(\bar{\beta}_1, \bar{\beta}_1)^\top, \ldots, ((\bar{\beta}_k, \bar{\beta}_k)^\top],$$

where $\bar{a}_i = a_i - a'_i$ and $\bar{\beta}_i = \beta_i \boxminus \beta'_i$ for $i \in \{1, \ldots, k\}$ using Add_1, Mult, and Add_2 of $\mathcal{HE}$. In this case the decryption of the level-2 ciphertext S, requires $(2k+1)$ $\widehat{\mathsf{Dec}}$ operations. However, using the Square operation and applying further optimization, we may simplify $S = (\alpha, \boldsymbol{\beta})$ of $\mathcal{HE}'$ as follows:

$$\alpha \leftarrow \widehat{\mathsf{Enc}}(\Sigma_{i=1}^{k}\bar{a}_i^2, pk) \boxplus (\boxplus_{i=1}^{k} 2\bar{a}_i \cdot \bar{\beta}_i),$$
$$\boldsymbol{\beta} \leftarrow [(\bar{\beta}_1, 0)^\top, \ldots, (\bar{\beta}_k, 0)^\top]. \tag{3}$$

Therefore, the decryption of S requires $(k+1)$ $\widehat{\mathsf{Dec}}$ operations using (2), which is almost twice as fast than for $\mathcal{HE}$. We define $\mathsf{ESED}(\boldsymbol{C}, \boldsymbol{C}', pk)$ as an algorithm that computes the encrypted SED $S = (\alpha, \boldsymbol{\beta})$ from $\boldsymbol{C}$ and $\boldsymbol{C}'$ using (3).

B. Proposed Privacy-Preserving Face Verification System

The proposed privacy-preserving face verification system is composed of two software modules: a face processing module and an authentication module. The face processing module detects and aligns the user's face image and extracts a feature vector from the aligned image using *ResNet*. The authentication module performs a privacy-preserving face verification protocol with the help of the face processing module. Specifically, the protocol has three phases: (1) setup phase, (2) registration phase, and (3) authentication phase. In the setup phase we set three "public parameters": a security parameter, a pre-trained face model, and a threshold of face verification to use *ResNet* which are shared by the client and server. In particular, the pre-trained face model is used by the face processing module to extract feature vectors on the client side. In the registration phase, the client generates an HE key pair and extracts the feature vector using the face processing module. Then, the client uses the public key to encrypt the feature vector and sends it to the server together with the client's public key and identifier. In the authentication phase, the client extracts and encrypts the feature vector in the same way as in the registration phase, and transmits it to the server together with the claimed identifier. Then, using secure 2PC, the server obtains a matching score, calculated using the stored feature vector and the input feature vector. In return, the client obtains a one-bit authentication result. The details of these three phases are as follows.

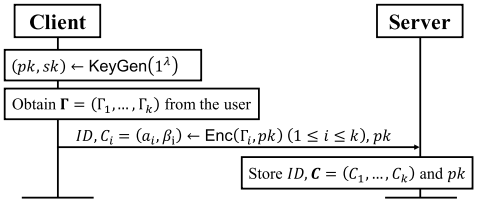

Fig. 2. Registration phase of the protocol.

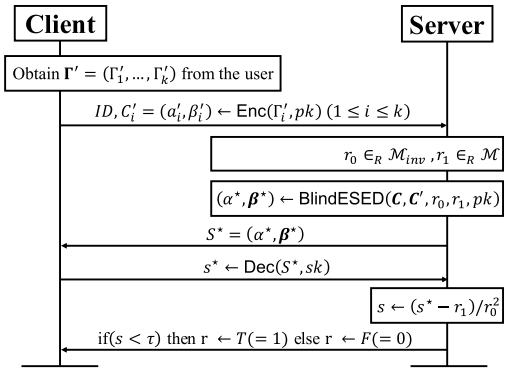

Fig. 3. Authentication phase of the protocol.

1) Setup Phase: The setup phase consists of setting: the security parameter λ, the pre-trained face model μ, and the threshold value τ, sharing them between the client and server. In particular, τ is defined according to the trained face model μ. If SED between two feature vectors extracted from two face images using the face model μ is smaller than τ, our system judges the owners of these two vectors as the same person.

2) Registration Phase: Fig. 2 shows the registration phase of our protocol. First, the client generates a key pair using KeyGen of $\mathcal{HE}'$ with the security parameter λ. The client obtains a k-dimensional feature vector $\boldsymbol{\Gamma} = (\Gamma_1, \ldots, \Gamma_k)$ extracted from a high dimensional face image of its user using the face processing module. To represent the elements of feature vectors in the encrypted domain, we multiply each element of the feature vector by 128 and cast it into an integer by rounding down. This quantization is required because the elements of a feature vector are rational numbers whereas the plaintext space of $\mathcal{HE}'$ is expressed only with integers. We denote by $\mathcal{F}$ the feature vector space with the quantization applied. According to our experiments, this quantized feature vector had essentially the same biometric accuracy as when the elements of the feature vector were expressed by rational numbers. Therefore, the face processing module extracts a feature vector $\boldsymbol{\Gamma} \in \mathcal{F}$. Next, the client encrypts $\boldsymbol{\Gamma}$ to a vector of level-1 ciphertexts, $\boldsymbol{C} = (C_1, \ldots, C_k)$, using the client's public key pk. Finally, the client sends the ciphertexts, public key, and its user's identifier to be stored in the server.

3) Authentication Phase: Fig. 3 shows the authentication phase. First, the client obtains a feature vector $\boldsymbol{\Gamma}' = (\Gamma'_1, \ldots, \Gamma'_k)$ extracted from a face image of the claimed user, using the face processing module in the same manner as the

registration phase, and encrypts $\mathbf{\Gamma}'$ to $\boldsymbol{C}' = (C'_1, \ldots, C'_k)$. The client sends to the server $\boldsymbol{C}'$, and its user's identifier. Now the server's goal is to compute the authentication score, i.e., SED, between the two encrypted feature vectors, $\boldsymbol{C}$ and $\boldsymbol{C}'$. However, the server does not perform directly ESED, but perform BlindESED, a blinded version of ESED. The role of BlindESED is to compute the encrypted value of SED between two encrypted vectors, without revealing the exact value to malicious clients. BlindESED takes as input $\boldsymbol{C} = ((a_1, \beta_1), \ldots, (a_k, \beta_k))$, $\boldsymbol{C}' = ((a'_1, \beta'_1), \ldots, (a'_k, \beta'_k))$, and pk, and two randomizers $r_0 \in_R \mathcal{M}_{inv}$ and $r_1 \in_R \mathcal{M}$, where $\mathcal{M}_{inv}$ is the set of all $m \in \mathcal{M}$ whose square m^2 has a multiplicative inverse. It outputs a level-2 ciphertext $S^\star = (\alpha^\star, \boldsymbol{\beta}^\star)$. To be precise,

$$S^\star = (\alpha^\star, \boldsymbol{\beta}^\star) \leftarrow \mathsf{BlindESED}(\boldsymbol{C}, \boldsymbol{C}', r_0, r_1, pk),$$

where $\alpha^\star$ and $\boldsymbol{\beta}^\star$ are calculated as follows:

$$\begin{aligned}\alpha^\star \leftarrow\ & r_0^2 \cdot \{\widehat{\mathsf{Enc}}(\Sigma_{i=1}^k (a_i - a'_i)^2, pk) \\ & \boxplus (\boxplus_{i=1}^k (2(a_i - a'_i) \cdot (\beta_i \boxminus \beta'_i)))\} \\ & \boxplus \widehat{\mathsf{Enc}}(r_1, pk), \qquad (4)\end{aligned}$$

$$\boldsymbol{\beta}^\star \leftarrow [(r_0 \cdot (\beta_1 \boxminus \beta'_1), 0)^\top, \ldots, (r_0 \cdot (\beta_k \boxminus \beta'_k), 0)^\top]. \qquad (5)$$

Next, the server sends the output $S^\star$ of BlindESED to the client, and the client decrypts it using (2) to acquire a blinded score $s^\star$. Although it holds that $s^\star = r_0^2 \times \mathsf{SED}(\mathbf{\Gamma}, \mathbf{\Gamma}') + r_1$ (See the correctness proof in the next section.), the client cannot obtain any information related to $\mathsf{SED}(\mathbf{\Gamma}, \mathbf{\Gamma}')$ owing to the randomizers r_0 and r_1 (See the security proof in the next section.). Next, when the client sends $s^\star$ to the server, the server calculates $(s^\star - r_1)/r_0^2$ to unblind $s^\star$ and obtains the authentication score s. The server sets the authentication result $\mathbf{r}$ to $T(= 1)$ if s is smaller than the threshold value τ; otherwise, the server sets it to $F(= 0)$. Finally, it sends $\mathbf{r}$ to the client. Throughout this paper, we refer to the authentication phase of our protocol as Π.

V. Analysis

A. Correctness

In this section, we prove that if the server and client follow the protocol, the server obtains SED in Π.

In Π, the server already has $\boldsymbol{C} = ((a_1, \beta_1), \ldots, (a_k, \beta_k))$ and obtains $\boldsymbol{C}' = ((a'_1, \beta'_1), \ldots, (a'_k, \beta'_k))$ from the client, where $(a_i, \beta_i) = (\Gamma_i - b_i, \widehat{\mathsf{Enc}}(b_i, pk))$, and $(a'_i, \beta'_i) = (\Gamma'_i - b'_i, \widehat{\mathsf{Enc}}(b'_i, pk))$. If we define $\bar{\Gamma}_i = \Gamma_i - \Gamma'_i$, $\bar{a}_i = a_i - a'_i$, and $\bar{b}_i = b_i - b'_i$, (4) can be rewritten as

$$\begin{aligned}\alpha^\star &= \widehat{\mathsf{Enc}}(r_0^2 \times \Sigma_{i=1}^k (\bar{a}_i^2 + 2\bar{a}_i \bar{b}_i) + r_1, pk) \\ &= \widehat{\mathsf{Enc}}(r_0^2 \times \Sigma_{i=1}^k ((\bar{\Gamma}_i - \bar{b}_i)^2 + 2(\bar{\Gamma}_i - \bar{b}_i)\bar{b}_i) + r_1, pk) \\ &= \widehat{\mathsf{Enc}}(r_0^2 \times \Sigma_{i=1}^k (\bar{\Gamma}_i^2 - \bar{b}_i^2) + r_1, pk).\end{aligned}$$

That is, $\alpha^\star$ is an $\widehat{\mathcal{HE}}$ ciphertext of $r_0^2 \times \Sigma_{i=1}^k (\bar{\Gamma}_i^2 - \bar{b}_i^2) + r_1$. Then, using (2), $s^\star \leftarrow \mathsf{Dec}(S^\star, sk)$ can be expressed as follows:

$$\begin{aligned}s^\star &\leftarrow \mathsf{Dec}((\alpha^\star, \boldsymbol{\beta}^\star), sk) \\ &= \widehat{\mathsf{Dec}}(\alpha^\star, sk) + \Sigma_{i=1}^k (\widehat{\mathsf{Dec}}(\beta^\star_{1,i}, sk))^2 \\ &= r_0^2 \times \{\Sigma_{i=1}^k (\bar{\Gamma}_i^2 - \bar{b}_i^2)\} + r_1 + r_0^2 \times \Sigma_{i=1}^k \bar{b}_i^2 \\ &= r_0^2 \times \Sigma_{i=1}^k \bar{\Gamma}_i^2 + r_1 \\ &= r_0^2 \times \mathsf{SED}(\mathbf{\Gamma}, \mathbf{\Gamma}') + r_1.\end{aligned}$$

Therefore, the server obtains $s = \mathsf{SED}(\mathbf{\Gamma}, \mathbf{\Gamma}')$ by computing $(s^\star - r_1)/r_0^2$.

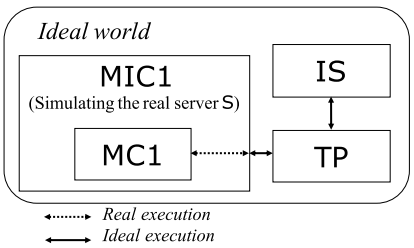

Fig. 4. Conceptual structure of the simulation proof for a 2PC protocol (TP: trusted third party, MC1: malicious client in real world, MIC1: malicious client in ideal world, S: server in real world, IS: server in ideal world)

B. Security

In this section, we analyze the security of Π against the following three adversaries: HBCS, MC1, and MC2.

1) Security Against HBCS*:* First, we define and analyze the security of Π against the adversary HBCS. It is clear that HBCS obtains an authentication score SED after participating in Π. After repeating the authentication sessions with the same client a sufficient number of times, HBCS may also acquire the distribution of SED. We want to guarantee that HBCS gains no more information than this score. We will say that Π is secure against HBCS if the execution of Π does not allow HBCS to obtain any information about the legitimate user's biometric data, that is, if Π is zero-knowledge. We show this in Theorem 2 under the obvious assumption that HBCS knows the distribution of SED. The proof is given in Appendix B.

Theorem 2: Π *is zero-knowledge for* HBCS.

2) Security Against MC1*:* We prove the security of Π against the adversary MC1 using a simulation proof. Simulation proof is a well-known proof technique that has been used commonly in the literature, e.g. [25], [38]. Fig. 4 shows the conceptual structure of a simulation proof for a 2PC protocol. A real execution means that each party conducts the protocol in the real world. On the other hand, an ideal execution in an ideal world means that two ideal parties, i.e., the ideal client and server, forward their input to a trusted third party (TP), and then TP calculates and returns the normal output of each party to the corresponding party. The proof shows that an adversary cannot distinguish between a real execution and an ideal execution. In Fig. 4, MIC1, a malicious client in the ideal world, simulates a real server S using only two values: the input to the client and TP's output. In other words, MIC1 is given only what a client is supposed to have in a normal execution of the protocol, and plays the role of S, interacting with MC1. Therefore, if MC1 cannot distinguish between the real execution of S and the simulation of MIC1, this means

that MC1 cannot obtain any information except what it should have in a normal execution, even if MC1 manipulates the real protocol.

Now we proceed to the proof of security against MC1. Let $Y = \mathbf{\Gamma}' = (\Gamma'_1, \ldots, \Gamma'_k) \in \mathcal{F}$, and $Z = (\mathsf{Enc}(\Gamma_1, pk), \ldots, \mathsf{Enc}(\Gamma_k, pk)) \in (\mathcal{M} \times \widehat{\mathcal{C}})^k$ be the inputs to client C and server S in the real world, respectively. For simplicity, we omit ID and pk in our notation. Then, the real-world execution of Π can be written as follows:

$$Real^{\Pi}_{\mathsf{C},\mathsf{S}}(Y, Z) \rightarrow (\mathbf{r}, s),$$

where $\mathbf{r}$, an authentication result, is the output of C, and s, an authentication score, is the output of S. Next, we define the ideal protocol $\mathsf{I}\Pi$ in the ideal world. In an ideal execution of $\mathsf{I}\Pi$, a client IC and a server IS forward their inputs Y and Z to TP, and then TP returns the outputs $\mathbf{r}$ and s to IC and IS, respectively. This execution using the same inputs and outputs as the real execution is denoted as follows:

$$Ideal^{\mathsf{I}\Pi}_{\mathsf{IC},\mathsf{IS}}(Y, Z) \rightarrow (\mathbf{r}, s).$$

Now, the security of Π against MC1 is defined as follows.

Definition 1 (Private Matching Protocol for MC1*):* Let S and IS be honest servers for the protocol Π in the real and ideal worlds, respectively. We say that Π is a private matching protocol for MC1 if for every PPT adversary MC1 there exists a PPT adversary MIC1 such that for all Y, Z:

$$Ideal^{\mathsf{I}\Pi}_{\mathsf{MIC1},\mathsf{IS}}(Y, Z) \stackrel{c}{\equiv} Real^{\Pi}_{\mathsf{MC1},\mathsf{S}}(Y, Z),$$

where $\stackrel{c}{\equiv}$ denotes that two distributions are computationally indistinguishable.

The following theorem shows that Π is a private matching protocol for MC1. That is, the execution of Π does not allow MC1 to obtain any information except $\mathbf{r}$. The proof is given in Appendix C.

Theorem 3: For every PPT adversary MC1 *to* Π*, there exists a PPT adversary* MIC1 *such that for all* Y, Z*:*

$$Ideal^{\mathsf{I}\Pi}_{\mathsf{MIC1},\mathsf{IS}}(Y, Z) \stackrel{c}{\equiv} Real^{\Pi}_{\mathsf{MC1},\mathsf{S}}(Y, Z).$$

3) Security Against MC2*:* Now we analyze the security of Π against the adversary MC2. In the previous subsection, we proved that Π is a private matching protocol, that is, there is no additional information on the user's biometric data that a malicious client can obtain from $S^\star$ and r in Π. Therefore, the only way that MC2 can be successfully authenticated is to forge its messages to the server without the user's biometric data. Namely, MC2 may try to forge its messages so that $s^\star$ satisfies $0 \leq (s^\star - r_1)/r_0^2 < \tau$. In Π, the messages sent by the client are ID, $\boldsymbol{C}'$, and $s^\star$. Because it is meaningless to forge ID, MC2 can choose between two strategies. First, it may try to forge the encrypted feature vector $\boldsymbol{C}'$, expecting that $\boldsymbol{C}'$ should be close from $\boldsymbol{C}$. On the other hand, MC2 may also try to forge the blinded score $s^\star$ itself, regardless of $\boldsymbol{C}'$.

To quantify the success probability of MC2, let $\mathbf{c}$ be the probability that a randomly generated feature vector $\mathbf{\Gamma}'$ may satisfy $\mathsf{SED}(\mathbf{\Gamma}, \mathbf{\Gamma}') < \tau$. This probability should be very small for a biometric authentication method to guarantee enough security. In addition, there could be other possibilities that an adversary may pass the authentication test without generating a proper feature vector $\mathbf{\Gamma}'$, such as forging $s^\star$ in Π. For secure authentication, this possibility should also be very small. Overall, it is desirable that the probability of an unauthorized user passing the authentication test should be small, but obviously it cannot be smaller than $\mathbf{c}$. The following definition reflects this observation, and states that an adversary to Π without any knowledge of $\mathbf{\Gamma}$ should have only a negligible amount of advantage over a random-guessing attacker.

Definition 2: Let $\mathbf{c}$ be the probability that a feature vector $\mathbf{\Gamma}'$ that was generated randomly without any knowledge of $\mathbf{\Gamma}$ may satisfy $\mathsf{SED}(\mathbf{\Gamma}, \mathbf{\Gamma}') < \tau$, where τ is a threshold. Let P be the probability that MC2 may be authenticated without any knowledge of $\mathbf{\Gamma}$. If $P \leq \mathbf{c} + O(2^{-\mathcal{L}})$ for a sufficiently large $\mathcal{L}$, the biometric authentication protocol is secure against MC2.

To show that Π is secure against MC2, we define a property regarding the underlying LHE.

Definition 3 (Uniform Encryption): Let $\widehat{\mathcal{HE}} = (\widehat{\mathsf{KeyGen}}, \widehat{\mathsf{Enc}}, \widehat{\mathsf{Eval}}, \widehat{\mathsf{Dec}})$ be an LHE scheme. Let $\mathcal{C}_m \subset \mathcal{C}$ be the ciphertext set corresponding to plaintext m. $\widehat{\mathcal{HE}}$ is a uniform encryption scheme if the cardinality of $\mathcal{C}_m$ is constant for all $m \in \mathcal{M}$.

Theorem 4: Assume that the underlying LHE of Π *is a uniform encryption scheme. Let* $\mathsf{forge}(\boldsymbol{C}')$ *and* $\mathsf{forge}(s^\star)$ *be the events that* MC2 *successfully forges* $\boldsymbol{C}'$ *and* $s^\star$ *in* Π*, respectively. Then,* $\Pr[\mathsf{forge}(\boldsymbol{C}') \vee \mathsf{forge}(s^\star)] \leq \mathbf{c} + \frac{\tau}{|\mathcal{M}|}$.

Most LHE schemes, including the Joye-Libert LHE [20] considered in this paper, satisfy Def. 3. In addition, in a typical setting where Euclidean-distance-based matching is used, $\frac{\tau}{|\mathcal{M}|}$ is very small. For example, $|\mathcal{M}| = 2^{96}$ for [20] with security parameter $\lambda = 80$. In general, the threshold τ is significantly less than this. For example, according to the experimental results explained in the next section, $\tau \leq 2^{12.6}$. As a result, $\mathcal{L} > 80$. Therefore, Π is secure against MC2. The proof of the theorem is given in Appendix D.

VI. Performance Evaluation

To verify the feasibility of the proposed method, we implemented a face verification system using well-known image processing techniques together with the proposed protocol. The system consists of a mobile client and an authentication server. The mobile client runs on a Samsung Galaxy S8+ device with Android 8.0, 6 GB of main memory and 128 GB of storage, and the server runs on a desktop computer with Intel Core i7-7700K CPU @ 4.20GHz and 32 GB of main memory. The proposed $\mathcal{HE}'$ as well as its underlying LHE scheme [20] were implemented as a C++ library using the GNU Multiple Precision Arithmetic Library (GMP) 6.1.2 [52] both in the client and server. We set the security parameter λ of [20] to 80. The face processing module was implemented as a C++ library using OpenCV 3.4.1 [53] and Dlib 19.16 [54]. In addition, we used OpenBLAS 0.2.20 [55], an optimized Basic Linear Algebra Subprograms (BLAS) library, to optimize linear algebraic operations of the face processing module. We applied our C++ libraries to the mobile client with Java Native Interface (JNI). The authentication server runs a web application written in C++, implemented using our $\mathcal{HE}'$ library.

The most time-consuming part in the proposed system is the level-2 decryption for $\mathcal{HE}'$. To accelerate this operation, we

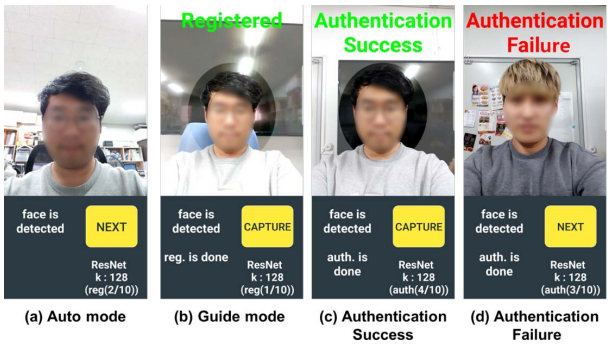

Fig. 5. Screenshots of our mobile application.

applied parallelization. According to (2), the level-2 decryption procedure of $\mathcal{HE}'$ consists of $l+1$ $\widehat{\text{Dec}}$s, and each $\widehat{\text{Dec}}$ is independent. We could perform these $\widehat{\text{Dec}}$ operations in parallel using Open Multi-Processing (OpenMP) 4.5 [46].

Regarding the face model μ of the face processing module in our system, we used the pre-trained model [56] provided by Dlib. The face model has been trained from a dataset of about 3 million faces of 7,485 people that were collected from multiple public face datasets and the Internet. The face model includes the layers and optimized weights for *ResNet*. In this model, the dimension k of a feature vector, is 128.

A. Design of Experiment

The feasibility of our system was verified by an experiment involving real users in a real environment. In the experiment, we used two face input modes: *Auto* and *Guide*. When the *Auto* mode is used, the face processing module automatically detects the face region from an image taken by the smartphone camera. The *Guide* mode provides a guideline on the smartphone screen so that a user may fit it to his/her face to manually capture a face image. The purpose of considering two separate modes was to see if the face data input method would affect performance and usability. In general, it is expected that users prefer an automatic input method because it does not require any specific action. However, the alignment results of the automatically captured face images might not be as good as those that were manually aligned using an explicit guideline. Therefore, we attempted to evaluate the performance and usability of the two face input modes by using the participants' feedback and the biometric accuracy obtained from the experiment.

Each participant repeats the registration and authentication phases of our system 10 times for each face input mode. Fig. 5 shows the screenshots of our mobile application used in the experiment. Fig. 5(a) and Fig. 5(b) are the user interfaces for the *Auto* and *Guide* modes, respectively, used to take the face image for registration and authentication. Fig. 5(c) shows a screenshot when the person that registered in (b) succeeds in authentication. Fig. 5(d) shows a screenshot when a person, different from the one who registered in (b) fails in authentication.

We remark that the decision of either success or failure should be made according to an appropriate threshold value τ of the SED between the feature vectors extracted from the registered and authenticated face images. Because the optimal τ that may guarantee the best equal error rate (EER) for real users was not yet available at the design stage, we set a candidate τ through a pilot study where we tested the pre-trained face model μ with the ORL dataset [57]. Its purpose was to provide participants with realistic feedback as close as possible to real environments. The value of τ was later updated after analyzing our experimental results explained in Section VI-D. The difference between the success rates using the conjectured and optimal τ was approximately 2%.

Fig. 6. Breakdown of the proposed face verification procedure.

The experiment was conducted in a controlled laboratory to avoid distraction. However, to make the experiment as realistic as possible, we allowed participants to perform actions that can be done during normal face authentication, such as adjusting the illumination, camera angle, and distance between the face and smartphone, wearing glasses, changing the hair style, and even partially covering the face with their hands.

Our experiments were performed in the following order for each participant.

1) The academic motivation and detailed procedure of the experiment are explained to the participant and the participant signs an informed consent form.
2) The participant selects the *Guide* mode, repeats the registration phase 10 times, and then repeats the authentication phase 10 times.
3) The participant selects the *Auto* mode, repeats the registration phase 10 times, and then repeats the authentication phase 10 times.
4) The participant responds to the follow-up questionnaire.

When conducting the experiment, we tried to ensure that possible ethical issues were properly handled. The above informed consent form was to inform the participants of the collection and usage of their personal information and the details about the experiment, and gain the consent of the participants, conforming to the Personal Information Protection Act [58] of Korea. In particular, the face images were used only for the specified purpose, i.e., measurement of the execution time and biometric accuracy, maintained in encrypted form, and were not disclosed anywhere.

B. Participants

We recruited 30 volunteers from our university campus. The participants were Asians with an average age of 25, consisting of 24 males (80%) and 6 females (20%). Of these participants, 27 people (90%) were using biometric authentication in their daily lives, 27 (90%) were right-handed and 17 (56.7%) were wearing glasses. We provided a reward equivalent to $ 9 to each participant in the experiment.

C. Execution Time

Fig. 6 shows a breakdown of the proposed face verification procedure, where prefixes C and S stand for the execution on the client and server sides, respectively. C1 covers all face processing procedures including face detection, alignment and

TABLE I

EXECUTION TIME OF EACH STEP AND COMMUNICATION OVERHEAD IN EACH PHASE OF THE PROPOSED FACE VERIFICATION SYSTEM (MS)

Dataset / Mode	#measurements	Authentication								(Offline) Registration	
		(C1)	(C2)	(S1)	(C3)	(S2)	Network	Total Time	Com.[1] (KiB)	Total Time	Com.[1] (KiB)
Auto	300	193.27	49.40	13.53	963.14	0.01	80.38	1299.73	33.64	349.68	17.76
Guide	300	157.47	45.63	13.52	931.45	0.01	79.83	1227.91		307.04	
CFP	2,471	150.15	44.76	13.63	789.34	0.01	76.84	1074.72	33.64	292.02	17.76
ORL	197	147.33	44.88	13.58	765.55	0.01	73.94	1045.29		290.28	

[1] Amount of transmitted data ($(268k + 140)$ bytes for authentication phase and $(140k + 264)$ bytes for registration phase when $\lambda = 80$)

feature extraction methods using *ResNet*. The remaining four steps from C2 to S2 correspond to Π. C2 is the task for encrypting the extracted feature vector $\mathbf{\Gamma}'$. S1 is the server's task for choosing arbitrary randomizers r_0, r_1 and performing BlindESED to compute the blinded value of ESED. C3 is the task for decrypting $S^\star$ to obtain $s^\star$. Finally, S2 is the task for unblinding the blinded value $s^\star$ to determine the result $\mathbf{r}$.

Table I presents the measured execution time and network overhead of the proposed system. As explained before, the performance was measured for $\lambda = 80$ and $k = 128$. The figures in the table are the 15% trimmed means, and the "total time" includes the communication and network delay, which is presented in the "Network" column. The two top rows in Table I present the results of the experiment in the real environment explained in Section VI-A and Section VI-B. The column named "#measurements" presents the total number of measurements. For example, because each of the 30 participants repeated the registration phase 10 times and the authentication phase 10 times for the *Audo* mode, we have 300 measured values for each phase of the *Auto* mode. The same holds for the *Guide* mode. To provide more objective numerical data on the performance of the proposed method, we also present the experimental results when the public face datasets *CFP* [59] and *ORL* [57] were applied to our system instead of the real participants' face images. The two bottom rows in the table present those results.

According to Table I, all privacy-preserving operations in the authentication phase (C2 through S2) were completed in approximately 1 s. Even when we counted both image processing time (for C1) and transmission time, a face verification task took less than 1.3 s at most. This demonstrates that biometric authentication can be performed efficiently and securely using the proposed method.

Now we analyze the experimental results more precisely. According to Table I, the most time-consuming operation in the authentication phase was still C3, i.e., $\mathcal{HE}'$ decryption. It would have taken significantly longer if C3 were not optimized by the improved transformation and parallelization. The *Guide* mode performed slightly faster than the *Auto* mode because additional sub-procedures were required to detect faces from the input images for the *Auto* mode. The face verification for the *CFP* and *ORL* datasets were faster than those in the real environment, because no threads continuously refreshing the screen with camera images were required, and all cores were assigned to parallel decryption threads during the C3 step.

Regarding the communication overhead, the amount of transmitted data was small, and the total transmission time for each phase was less than 100 ms in all datasets in Table I.

Finally, we summarize the participants' replies to the follow-up questionnaire. 22 (73.3%) among the 30 participants thought that the proposed system was quite fast that they did not feel any delay. 27 (90%) preferred the *Auto* mode, which is consistent with our conjecture that users generally prefer an automatic method. 5 (17%) answered that they would use the *Auto* mode even if its accuracy could be lower than that of the *Guide* mode.

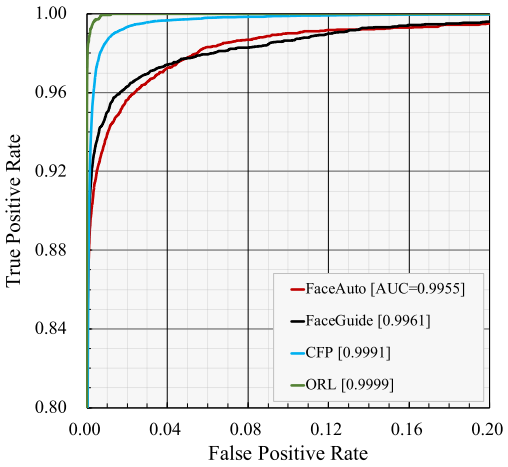

Fig. 7. ROC curves for the *FaceAuto*, *FaceGudie*, *CFP*, and *ORL* datasets.

D. Biometric Accuracy

As the metric for accuracy, we used EER and the area under the curve (AUC) of the receiver operating characteristic (ROC). The indicator EER is the most commonly used for biometric accuracy and is the point at which the false acceptance rate (FAR) curve and the false rejection rate (FRR) curve cross each other, where FAR is the probability that a biometric system will incorrectly accept an access attempt of an unauthorized user, and FRR is the probability that an authorized user will be incorrectly rejected. The threshold value τ corresponding to the crossing point, i.e, FAR-FRR equilibrium, is called an optimal threshold. The AUC metric is well-known metric for the predicted quality of the classification model in machine learning [60], and is calculated as the area under the ROC curve that plots two parameters, False Positive Rate = FAR and True Positive Rate = 1 − FRR. In general, a biometric system with low EER and high AUC is considered an accurate system.

Table II presents the EER, AUC, and the optimal threshold τ for the four datasets, and Fig. 7 shows the ROC curves. For readability, we zoomed in on the curves and only showed the region for the TPR between 0.8 and 1.0 and the FPR between 0 and 0.2. *FaceAuto* and *FaceGuide* represent the face datasets obtained through our experiment for real users using the *Auto*

TABLE II
BIOMETRIC ACCURACY OF THE PROPOSED FACE VERIFICATION SYSTEM

Dataset	#people	#faces	EER	AUC	Optimal τ
FaceAuto	30	600	3.27%	0.9955	$2^{11.4}$
FaceGuide	30	600	3.04%	0.9961	$2^{11.4}$
CFP	500	4,968	1.17%	0.9991	$2^{12.6}$
ORL	40	396	0.37%	0.9999	$2^{12.2}$

TABLE III
EXECUTION TIME AND COMMUNICATION OVERHEAD IN EACH PHASE OF THE PROPOSED SYSTEM WITH PCA (MS)

Mode	k	Authentication		(Offline) Registration	
		Total Time	Com.[1] (KiB)	Total Time	Com.[1] (KiB)
Auto	16	623.36	4.33	458.08	2.45
	24	705.55	6.42	462.69	3.54
	32	777.87	8.52	475.70	4.63
	40	848.94	10.61	480.54	5.73
	48	906.14	12.70	478.99	6.82
Guide	16	599.79	4.33	418.40	2.45
	24	665.51	6.42	417.81	3.54
	32	733.17	8.52	428.74	4.63
	40	798.57	10.61	429.58	5.73
	48	863.52	12.70	441.44	6.82

[1] Amount of transmitted data

and *Guide* modes, respectively. We also provide the biometric accuracy with the *CFP* and *ORL* datasets as an objective performance indicator for the proposed method. Although *CFP* and *ORL* contain 10 face images per person, a few face images were not successfully detected by Dlib. To measure EER and AUC, we only used the successfully detected face images. Table II presents the number of people and successfully detected faces in each dataset. The experimental results show that EERs for *FaceAuto* and *FaceGuide* are 3.27% and 3.04%, respectively, and AUCs are 0.9955 and 0.9961, respectively. It implies that the accuracy of our system using *Guide* is slightly better than that using *Auto*. This is because the face images obtained using *Auto* are less uniform in alignment, brightness, etc. than those obtained using *Guide*. In addition, the EERs of the proposed method using *CFP* and *ORL* are 1.17% and 0.37%, respectively. The reason why the EER and AUC of our method using public datasets are significantly better than those using face images from real participants is that the public datasets were composed of face images taken in a fairly uniform pose in a controlled environment.

E. Performance With PCA

Although our system was implemented using *ResNet*, it is also possible to use PCA if the user population is known. Therefore, we performed an additional experiment for our system, replacing *ResNet* with PCA. We tested both *Auto* and *Guide* modes and measured execution times with increasing k, the dimension of a feature vector, from 16 to 48 in the intervals of 8. The same 30 volunteers participated in this experiment, performing each phase (registration and authentication) 5 times for each k. The results in Table III are the 15% trimmed means for each k. According to the experimental results, the execution times of authentication with *Auto* and *Guide* are 623.36–906.14 ms and 599.79–863.52 ms, respectively, including the face processing and network overheads.

We also measured the EER and AUC of our PCA-based system. Of the 1,500 face images obtained for each face input mode, 1,050 images (70%) were used for PCA training and the remaining 450 images (30%) for testing. This is a well-known dataset configuration to measure biometric accuracy that has been used commonly in the existing literature related to PCA [61]. According to experimental results, EER and AUC of our PCA-based system for *FaceAuto* and *FaceGuide* were 6.20–8.10% and 0.9766–0.9846, and 4.50–6.12% and 0.9842–0.9905, respectively, for k between 16 and 48. In summary, when the user population is known in advance, the system using PCA can perform significantly faster than that using *ResNet*, with lower but reasonable biometric accuracy.

F. Comparison With Existing Methods

In this section, we compare the characteristics and performance of the proposed method with the existing 2PC-based methods. We do not consider the protocols in [37], [40], [41] that involve three or more parties. The methods shown in Table IV are classified into two categories of biometric authentication: biometric identification and biometric verification. The proposed protocol belongs to the latter. In all of the biometric identification methods listed in Table IV, the server stores n biometric data without encryption, and the client has the biometric data of only one user. The goal of these methods is to enable the server and client to perform biometric identification without providing their biometric data to each other. In addition, the matching result is not known to the server. On the other hand, in all of the verification methods listed in Table IV, the server possesses the encrypted biometric data of the corresponding registered user, whereas the client attempts authentication using the biometric data of a claimed user. These methods aim to prevent leaking to a client, in a decrypted form, the encrypted template stored in the server, and protect the input template from leaking to the server.

As demonstrated in the table, a notable property of our protocol is that it takes account of malicious clients. Most of the existing literature only consider honest-but-curious (HBC) clients. Obviously, an adversarial model with malicious clients is more realistic than those considering only HBC clients. It is not possible to directly compare our protocol with either the biometric identification methods or the biometric verification methods considering only HBC clients. Among the three verification methods that consider malicious clients, the method proposed in [38] adopts the absolute average deviation (AAD) as a matching function. Therefore, a direct comparison with it is not possible either. However, we see that the proposed method performs slightly faster than [38] in a typical setting. We also remark that the Euclidean distance metric is a more common approach to a matching function as shown in the table, in particular for face authentication. The only existing 2PC method that considers malicious clients and the Euclidean distance metric is [39], but it took 105.5 s to calculate the Euclidean distance. Finally, the method in [42] involves three parties as a whole, but requires only two parties in the authentication phase. This method is not directly comparable with ours, because its security relies in part on the use of TEE and password as well as biometrics. To remark on

TABLE IV
COMPARISON OF TWO-PARTY COMPUTATION-BASED PRIVACY-PRESERVING BIOMETRIC AUTHENTICATION METHODS

Type	Method	Biometric Data	Adversarial Model	Tools	Matching Function	Parameters	Speed
Biometric identification	[26]	face	HBC[1] server / HBC client	HE[2]	ED[3]	n=320, k=12	18.00s
	[27]	face	HBC server / HBC client	HE + GC[4]	ED	n=320, k=12	15.54s
	[28]	face	HBC server / HBC client	HE + OT[5]	HD[6]	n=100, k=900	31.00s
	[30]	fingerprint	HBC server / HBC client	HE	ED	n=320, k=16	16.00s
	[31]	fingerprint	HBC server / HBC client	HE + GC	ED	n=512, k=16	3.47s
	[32]	fingerprint or iris	HBC server / HBC client	HE + GC + OT	HD	n=100, k=900	20.00s
	[33]	not specified	HBC server / HBC client	ABY[7]	ED	n=512, k=4	0.49s
Biometric verification	[34]	fingerprint	HBC server / HBC client	GC	ED and HD	k=5	ED: 1.99s
	[35]	iris	HBC server / HBC client	HE	HD	k=2400	0.37s
	[36]	palmprint	HBC server / HBC client	HE	ED	k=100	15.88s
	[38]	implicit features	HBC server / malicious client	HE	AAD[8]	k=100	1.18s
	[39]	face	HBC server / malicious client	GC	ED and MD[9]	k=16	ED: 105.50s
	[42]	face	HBC server / malicious client	ZKPK[10] + TEE[11]	SVM[12]	k=60	15.42s
	Proposed	face	HBC server / malicious client	HE	ED	k=128	**1.07s**[†]

[1] Honest-But-Curious, [2] Homomorphic Encryption, [3] Euclidean Distance, [4] Garbled Circuit, [5] Oblivious Transfer, [6] Hamming Distance, [7] Arithmetic Sharing + Boolean Sharing + Yao's GC, [8] Absolute Average Deviation, [9] Manhattan Distance, [10] Zero-Knowledge Proof of Knowledge, [11] Trusted Execution Environment, [12] Support Vector Machine.

[†] For a fair comparison, we only counted the total execution time of Π including the network overheads, but excluding C1 for face processing tasks.

its performance only for reference, an authentication session in [42] took 15.42 s when $k = 60$. That is, our method was significantly faster than [39], [42] even though we set $k = 128$.

In the above comparison, we did not include the biometric authentication protocol in [43] because it requires a high communication cost for preprocessing. Most multiparty computation (MPC) schemes such as SPDZ [22], [23] require a preprocessing phase where input pairs and Beaver's triples are produced. We evaluated the communication costs for this phase of two-party Euclidean distance computation. We used two well-known tools for arithmetic circuit-based MPC protocols, MASCOT [62] and SPDZ2k [63] included in the MP-SPDZ project [64], which adopt various state-of-the-art optimization techniques. We measured the costs of computing the input pairs and square pairs in the preprocessing phase for a single authentication using 128-dimensional templates, where square pairs are the optimized versions of Beaver's triples for efficient squaring. According to our experimental results, the communication costs were 24.64 MB for MASCOT [62] and 21.19 MB for SPDZ2k [63]. Moreover, these pairs were not reusable for other authentication sessions. Thus, the communication cost for the preprocessing phase increased almost in proportion to υ, the number of authentications that will be performed. To be precise, it was $24.64\lceil \upsilon/8 \rceil$ MB and $21.19\lceil \upsilon/8 \rceil$ MB for MASCOT and SPDZ2k, respectively, when the default setting of MP-SPDZ was used. Therefore, the protocol proposed in [43] is less practical for mobile services such as banking and credit card approval, where authentication is expected to be performed frequently.

Finally, the remote authentication approach of the proposed system has several advantages over local authentication in terms of security and performance. For local authentication, transformation techniques such as cancelable biometrics [3], [4] have been frequently used to keep the biometric template secure even when the local system has been compromised. These techniques aim to transform the biometric template into another form such that the transformed template may not be directly used to recover the original template. This approach can be viewed as a blurring of the original data and, thus, inevitably sacrifices biometric accuracy.[1] On the contrary, the remote authentication approach keeps the quality of the original data, distributing the important data to two parties. In particular, the proposed system stores the encrypted data on a remote server and the key in the local client. The only concern is that the authentication may not be possible if the server is not available. However, considering that biometric authentication is frequently used for applications involving remote services such as banking, credit card approval and shopping, this stability issue can be solved by piggybacking the biometric authentication on the availability of these services. That is, we may let the service provider also play the role of an authentication server. Furthermore, the service providers may prefer to perform the authentication by themselves, rather than relying on local authentication.

VII. LIMITATIONS AND EXTENSION

The proposed system does not explicitly verify the validity of a user's biometric data. That is, the server cannot prevent a client from encrypting some garbage values instead of real biometric features and registering the result at the registration phase. However, this malicious client action can be easily prevented at the authentication phase if the legitimate user's data have been validated at the registration phase; simply, the authentication will not succeed. In some applications, it may be desirable to assure the server that the registration is actually based on the user's biometric identity, not a garbage value. The authentication protocol in [42] provides this capability using a trusted third party (TP). By introducing TP only to the registration phase in a similar way to [42], we add the capability of providing this assurance to our protocol. With TP, the registration phase of our protocol can be modified as

[1] According to the ROC curves in Fig. 7 in [4], the genuine accept rates of cancelable transforms were lower than those of the original template for almost all values of FAR except only a few regions where FAR $< 10^{-3}$.

TABLE V
EXECUTION TIME OF ADDITIONAL TASKS FOR EXTENDED PROTOCOL (MS)

Task	Execution Time
$\mathsf{Sign}(ID\|\|C_1\|\|\cdots\|\|C_k\|\|pk, sk_{sig})^{\dagger}$	0.39
$\mathsf{Verify}(ID\|\|C_1\|\|\cdots\|\|C_k\|\|pk, \sigma, pk_{sig})^{\ddagger}$	0.42

† Sign is the signing procedure of ECDSA
‡ Verify is the signature verification procedure of ECDSA.

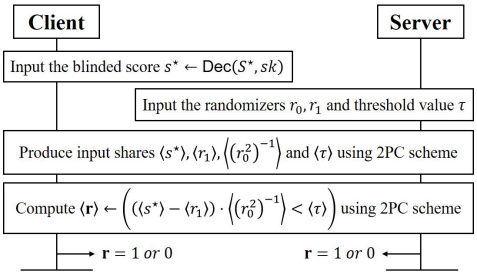

Fig. 8. Modification of Π for secure comparison (Sub-protocol for secure comparison). $\langle a\rangle$ is the share of secret value a.

follows, where pk_{sig} and sk_{sig} are TP's public and private keys for a digital signature scheme, respectively.

1) The client generates an $\mathcal{HE}'$ key pair (pk, sk), and obtains a feature vector $\mathbf{\Gamma} = (\Gamma_1, \ldots, \Gamma_k)$.
2) The client sends pk, $\mathbf{\Gamma}$ and the user's identifier ID to TP.
3) After verifying that $\mathbf{\Gamma}$ is valid, TP encrypts $\mathbf{\Gamma}$ to $\boldsymbol{C} = (C_1, \ldots, C_k)$ using pk, and signs $(ID||C_1||\cdots||C_k||pk)$ using sk_{sig}. Let σ be the resulting signature.
4) TP sends $\boldsymbol{C}$ and σ to the client.
5) The client sends ID, $\boldsymbol{C}$, pk, and σ to the server.
6) The server verifies σ using TP's public key pk_{sig} and stores ID, $\boldsymbol{C}$, pk, and σ. (We assume that the server already has pk_{sig}.)

The authentication phase does not change. Because the above modified registration phase ensures that the server receives and stores the ciphertext of a user's valid biometric data, we do not need any explicit validation procedure for the authentication phase. Even though no signature on $\boldsymbol{C}'$ is provided to the server, the encryption of garbage values will result in authentication failure, and will be rejected.

To verify the feasibility of this extension, we implemented a prototype system for TP on a desktop computer with Intel Core i7-7700K CPU @ 4.20GHz and 32 GB of main memory, and modified the relevant part in the client and server. For signature generation and verification, we ported the OpenSSL library [65] with ECDSA [66] into both TP and the server. We measured the performance of the modified tasks and calculated the increase in the amount of network traffic. Table V shows the execution time of additional tasks for the extended protocol. The figures in the table are the 15% trimmed means after 10,000 iterations when the NIST P-256 elliptic curve [66] was used for ECDSA. According to Table V, the sum of the execution time for the additional tasks is 0.81 ms, where TP and the server perform Sign and Verify, respectively. The results were the same for four datasets. Furthermore, the time for encrypting $\mathbf{\Gamma}$, which is equivalent to C2 in Table I, was significantly reduced because it was moved from a mobile client to a desktop TP. The encryption of $\mathbf{\Gamma}$ took less than 10 ms in TP. Because the communication delay was negligible in the original experiment in Table I, we did not separately consider the communication delay in the new experiment. Meanwhile, the amount of additional network traffic between the client and TP was 17.88 KiB, and that between the client and server was only approximately 0.06 KiB. Therefore, the extended protocol is expected to provide the biometric data validation functionality with only a small overhead.

Finally, the proposed system involves a secure computation only for distance computations, but does not perform distance comparison in a secure manner. That is, the actual distance between the registered and authenticated templates is revealed to the server, although the server does not obtain any information about the individual templates. Many previous protocols such as [38], [39] adopted the same approach. For a security model where the distance distribution should not be known to the server, Π can be extended to support secure comparison. We outline a solution using secure 2PC techniques, where the sub-protocol depicted in Fig. 8 replaces the last part in the authentication phase. Instead of the client sending $s^\star$ and server recovering s, the client and server perform a secure comparison using a 2PC scheme that supports the computation of arithmetic circuits in a malicious adversary setting. To verify the feasibility of the solution, we implemented a prototype using MASCOT [62]. As in a typical 2PC protocol, the new sub-protocol is composed of two phases, i.e., preprocessing and online phases. Our experimental results show that the execution time of the online phase was only 2.5 ms on a desktop computer with Intel Core i7-7700K CPU @ 4.20GHz and 32 GB of main memory. However, the communication overhead for the preprocessing phase for 2PC was 47.21 MB, which is rather large for a mobile environment. Other SPDZ-based protocols such as [43] suffer from the same issue. Even Boolean circuit-based 2PC protocols have the same issue. According to our experiments, the adapted SPDZ2k protocol included in MP-SPDZ [64], which is a Boolean circuit-based MPC protocol, also consumed 38.5 MB for the preprocessing phase. Therefore, it will be desirable to develop a secure comparison protocol with low communication overhead in a malicious adversary setting. This is out of the scope of this paper, and we will leave this for future research.

VIII. CONCLUSION

In this paper, we proposed a practical privacy-preserving 2PC-based face verification system that guarantees security against both, an honest-but-curious server, and malicious clients. Furthermore, we verified the performance with an experiment involving real users in a real environment. The proposed system implements state-of-the-art face image processing techniques and computes the matching score between two encrypted feature vectors using homomorphic encryption. For matching score computation, we adopted the (squared) Euclidean distance metric, which is one of the most universal matching functions in biometric authentication. For efficient computation on smartphones, we improved the Catalano-Fiore transformation [44], [45] which converts a linear homomorphic encryption scheme into a quadratic

scheme, and then parallelized the decryption of the improved transformation using OpenMP [46]. Owing to these optimizations, our protocol was significantly faster than the previous privacy-preserving authentication protocol [39], even though the dimension of feature vectors was greater. According to our experiments, the total execution time of all privacy-preserving operations in the authentication phase was approximately 1 s, and secure face verification was completed within 1.3 s on a smartphone with a reasonable EER of 3.04% for real-world users.

APPENDIX A
PROOF OF THEOREM 1

Because the level-1 ciphertexts of $\mathcal{HE}'$ are exactly the same as those of $\mathcal{HE}$ in [45], it is clear that the theorem holds for level-1 ciphertexts. Next, recall that a term in a level-2 ciphertext of $\mathcal{HE}'$ has the form of $(\widehat{\mathsf{Enc}}(m^2 - b^2, pk), (\widehat{\mathsf{Enc}}(b, pk), 0)^\top)$ for a plaintext m and a random plaintext b. Then, the semantic security of the level-2 ciphertexts can be proven in a straightforward fashion using the same technique as the proof in Section 4 of [45], as follows:

$$\begin{aligned}
&(\widehat{\mathsf{Enc}}(m_0^2 - b^2, pk), (\widehat{\mathsf{Enc}}(b, pk), 0)^\top) \\
&\quad \stackrel{c}{\equiv} (\widehat{\mathsf{Enc}}(m_0^2 - b^2, pk), (\widehat{\mathsf{Enc}}(0, pk), 0)^\top) \\
&\quad \equiv (\widehat{\mathsf{Enc}}(m_1^2 - b^2, pk), (\widehat{\mathsf{Enc}}(0, pk), 0)^\top) \\
&\quad \stackrel{c}{\equiv} (\widehat{\mathsf{Enc}}(m_1^2 - b^2, pk), (\widehat{\mathsf{Enc}}(b, pk), 0)^\top),
\end{aligned}$$

where $\stackrel{c}{\equiv}$ denotes that the two distributions are computationally indistinguishable by the semantic security of $\widehat{\mathcal{HE}}$, and $\equiv$ denotes that the distributions are identical. □

APPENDIX B
PROOF OF THEOREM 2

We will prove that HBCS can generate the exact distribution of transcripts of Π without participating in Π. We first provide the proof under the assumption that SED s is always zero for a legitimate user, and extend the proof to the case where s is not guaranteed to be 0 but its distribution is known.

A transcript T of Π (except ID) can be written as a tuple $T = (\boldsymbol{C}', \alpha^\star, \boldsymbol{\beta}^\star, s^\star, \mathbf{r})$, where $\boldsymbol{C}' = ((a'_1, \beta'_1), \ldots, (a'_k, \beta'_k))$. If an authentication is successful in Π, the distribution of T will be as follows: First, $\boldsymbol{a}' = (a'_1, \ldots, a'_k)$ is a random element in $(\mathcal{M})^k$ because $\boldsymbol{a}' = \boldsymbol{\Gamma}' - \boldsymbol{b}' \in (\mathcal{M})^k$ for a random vector $\boldsymbol{b}' = (b'_1, \ldots, b'_k) \in_R (\mathcal{M})^k$. If we denote the ciphertext set corresponding to a plaintext m by $\mathcal{C}_m \subset \mathcal{C}$, then $\boldsymbol{\beta}' = (\beta'_1, \ldots, \beta'_k)$ is a random element in $\mathcal{C}_{(\Gamma'_1 - a'_1)} \times \cdots \times \mathcal{C}_{(\Gamma'_k - a'_k)}$ according to the above relationship between $\boldsymbol{a}'$ and $\boldsymbol{b}'$. Next, the ciphertext $S^\star = (\alpha^\star, \boldsymbol{\beta}^\star)$ is determined by BlindESED from the input of: two ciphertexts $\boldsymbol{C} = ((a_1, \beta_1), \ldots, (a_k, \beta_k))$, $\boldsymbol{C}' = ((a'_1, \beta'_1), \ldots, (a'_k, \beta'_k))$, two randomizers $r_0 \in_R \mathcal{M}_{inv}$, $r_1 \in_R \mathcal{M}$, and the client's public key pk. Because s is always zero, $s^\star = r_1$ and $\mathbf{r} = 1$. As a result, the set of transcripts is given as follows with the distribution generated by the proper execution of Π:

$$\begin{aligned}
\mathcal{T} = \{&(\boldsymbol{C}', \alpha^\star, \boldsymbol{\beta}^\star, s^\star, \mathbf{r}) : \\
&\boldsymbol{C}' = ((a'_1, \beta'_1) \ldots, (a'_k, \beta'_k)) \\
&\in_R (\mathcal{M} \times \mathcal{C}_{(\Gamma'_1 - a'_1)}) \times \cdots \times (\mathcal{M} \times \mathcal{C}_{(\Gamma'_k - a'_k)}), \\
&r_0 \in_R \mathcal{M}_{inv}\ , r_1 \in_R \mathcal{M}, \\
&(\alpha^\star, \boldsymbol{\beta}^\star) \leftarrow \mathsf{BlindESED}(\boldsymbol{C}, \boldsymbol{C}', r_0, r_1, pk), \\
&s^\star \leftarrow r_1, \mathbf{r} \leftarrow 1\}.
\end{aligned}$$

Now, we show that HBCS alone can simulate $\mathcal{T}$ with the same distribution as above. Note that a transcript T is determined by $\boldsymbol{C}'$, r_0, and r_1 because $\boldsymbol{C}$ and pk are fixed. In addition, r_0 and r_1 are generated from a uniform distribution, which is easy to simulate. Therefore, the core of the proof is to show that HBCS can simulate the distribution of $\boldsymbol{C}'$. Given $\boldsymbol{C} = ((a_1, \beta_1), \ldots, (a_k, \beta_k))$ and pk, the simulator for HBCS performs the following procedure:

1) Choose $\boldsymbol{a}' = (a'_1, \ldots, a'_k) \in_R (\mathcal{M})^k$.
2) Set $\beta'_i \leftarrow \beta_i \boxplus \widehat{\mathsf{Enc}}(a_i - a'_i, pk)$ (where $1 \le i \le k$), and set $\boldsymbol{C}' = ((a'_1, \beta'_1), \ldots, (a'_k, \beta'_k))$.
3) Choose randomizers $r_0 \in_R \mathcal{M}_{inv}$ and $r_1 \in_R \mathcal{M}$.
4) $(\alpha^\star, \boldsymbol{\beta}^\star) \leftarrow \mathsf{BlindESED}(\boldsymbol{C}, \boldsymbol{C}', r_0, r_1, pk)$.
5) $s^\star \leftarrow r_1$.
6) $\mathbf{r} \leftarrow 1$.
7) Output $(\boldsymbol{C}', \alpha^\star, \boldsymbol{\beta}^\star, s^\star, \mathbf{r})$.

We see that β'_i computed in step 2 satisfies

$$\begin{aligned}
\beta'_i &= \widehat{\mathsf{Enc}}(b_i + (\Gamma_i - b_i) - a'_i, pk) \\
&= \widehat{\mathsf{Enc}}(\Gamma_i - a'_i, pk),
\end{aligned}$$

for $1 \le i \le k$. Because we assumed $s = 0$, i.e., $\boldsymbol{\Gamma}' = \boldsymbol{\Gamma}$, the vector $\boldsymbol{C}'$ generated in step 2 is a valid level-1 $\mathcal{HE}$ ciphertext vector for the feature vector $\boldsymbol{\Gamma}'$, even though HBCS does not know the values of Γ'_i. It is obvious that the remaining steps of the simulator are the same as those of Π. Consequently, the set of 5-tuples produced by the simulator and their probability distribution are exactly the same as those of $\mathcal{T}$. This implies that HBCS successfully simulates the distribution of the transcripts for Π without any interaction with the real client. This proves Theorem 2 for the case where $s = 0$.

Next, we consider the case where s is not guaranteed to be 0. In this case, the feature vector $\boldsymbol{\Gamma}'$ extracted from a legitimate user's face image is expected to be close to $\boldsymbol{\Gamma}$, the registered feature vector, even if they are not identical. We define $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_k) = \boldsymbol{\Gamma} - \boldsymbol{\Gamma}'$ and denote the distribution of $s = \mathsf{SED}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}')$ as $\mathcal{D}(\mathsf{SED}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}'))$. Note that $\boldsymbol{\delta}$ satisfies $s = ||\boldsymbol{\delta}||^2$, where $||\boldsymbol{v}||$ is a Euclidean norm of the vector $\boldsymbol{v}$. $\mathcal{D}(\mathsf{SED}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}'))$ is known to HBCS although HBCS does not know $\boldsymbol{\Gamma}$ nor $\boldsymbol{\Gamma}'$.

To generalize the proof to include the case of nonzero $\boldsymbol{\delta}$'s, we slightly modify steps 1, 5, and 6 of the simulator for HBCS as follows:

1) Choose $(a'_1, \ldots, a'_k) \in_R (\mathcal{M})^k$ and set $a''_i \leftarrow a'_i + \delta_i$ for $1 \le i \le k$, where $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_k)$ is randomly chosen such that $s = ||\boldsymbol{\delta}||^2$ for $s \in_R \mathcal{D}(\mathsf{SED}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}'))$.
2) Set $\beta'_i \leftarrow \beta_i \boxplus \widehat{\mathsf{Enc}}(a_i - a''_i, pk)$ (where $1 \le i \le k$), and set $\boldsymbol{C}' = ((a'_1, \beta'_1), \ldots, (a'_k, \beta'_k))$.
3) Choose randomizers $r_0 \in_R \mathcal{M}_{inv}$ and $r_1 \in_R \mathcal{M}$.
4) $(\alpha^\star, \boldsymbol{\beta}^\star) \leftarrow \mathsf{BlindESED}(\boldsymbol{C}, \boldsymbol{C}', r_0, r_1, pk)$.
5) $s^\star \leftarrow r_0^2 \times s + r_1$.

6) If($s < \tau$) then $\mathbf{r} \leftarrow 1$; else $\mathbf{r} \leftarrow 0$.
7) Output $(\boldsymbol{C}', \alpha^\star, \boldsymbol{\beta}^\star, s^\star, \mathbf{r})$.

Step 1 was modified to reflect the distribution of $\mathsf{SED}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}')$. That is, a score s is chosen from $\mathcal{D}(\mathsf{SED}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}'))$, and the simulation of $\boldsymbol{C}'$ is done by conducting the operations in inverse order from s. Step 5 was also modified to cover a nonzero authentication score s. Step 6 was modified to reflect the fact that even for a legitimate user, s may be greater than the threshold τ, which results in an authentication failure. Now it is straightforward that the set of 5-tuples generated by the modified simulator and their probability distribution are the same as those of the proper execution of Π, which is

$$\begin{aligned}\overline{\mathcal{T}} = \{&(\boldsymbol{C}', \alpha^\star, \boldsymbol{\beta}^\star, s^\star, \mathbf{r}) : s \in_R \mathcal{D}(\mathsf{SED}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}')),\\ &\boldsymbol{\delta} = (\delta_1, \ldots, \delta_k) \in_R \{\boldsymbol{\delta} \mid s = ||\boldsymbol{\delta}||^2\},\\ &\boldsymbol{C}' = ((a'_1, \beta'_1) \ldots, (a'_k, \beta'_k))\\ &\in_R (\mathcal{M} \times \mathcal{C}_{(\Gamma_1 - \delta_1 - a'_1)}) \times \cdots \times (\mathcal{M} \times \mathcal{C}_{(\Gamma_k - \delta_k - a'_k)}),\\ &r_0 \in_R \mathcal{M}_{inv}, r_1 \in_R \mathcal{M},\\ &(\alpha^\star, \boldsymbol{\beta}^\star) \leftarrow \mathsf{BlindESED}(\boldsymbol{C}, \boldsymbol{C}', r_0, r_1, pk),\\ &s^\star \leftarrow r_0^2 \times s + r_1, \mathbf{r} \leftarrow 1 \text{ if } s < \tau; \text{otherwise } \mathbf{r} \leftarrow 0\}.\end{aligned}$$

□

Appendix C
Proof of Theorem 3

In a nutshell, the tricky part in the construction of MIC1 is that MIC1 has to provide MC1 with $S^\star$, but this requires the knowledge of $\boldsymbol{C}$. Even though MIC1 has Γ' as input and it can receive $\mathbf{r}$ from TP, MIC1 does not know the original Γ nor its encrypted value $\boldsymbol{C}$. Therefore, instead of $\boldsymbol{C}$, MIC1 just uses $\boldsymbol{C}''$, which was conjectured from $\mathbf{r}$. Obviously, $\boldsymbol{C}''$ is different from $\boldsymbol{C}$, but MC1 cannot recognize this difference thanks to randomizers. To be precise, given a real-world PPT adversary MC1, the ideal world MIC1 is constructed as follows:

1) MIC1 forwards the input $Y = \Gamma'$ to TP and receives $\mathbf{r}$ from TP. (IS forwards Z to TP and receives s.)
2) MIC1 generates $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_k)$ according to $\mathbf{r}$.
   a) If $\mathbf{r} = 0$, then randomly chooses $\boldsymbol{\delta}$ such that $||\boldsymbol{\delta}||^2 = \tau$.
   b) If $\mathbf{r} = 1$, then $\boldsymbol{\delta} \leftarrow (0, \ldots, 0)$.
3) MIC1 forwards Y to MC1 and receives $\boldsymbol{C}'$.
4) MIC1 generates $\boldsymbol{b}'' = (b''_1, \ldots, b''_k) \in_R (\mathcal{M})^k$ and computes $\boldsymbol{C}''$ as follows:
$$\begin{aligned}\boldsymbol{C}'' \leftarrow\ &((\Gamma'_1 + \delta_1 - b''_1, \widehat{\mathsf{Enc}}(b''_1, pk)),\\ &\ldots, (\Gamma'_k + \delta_k - b''_k, \widehat{\mathsf{Enc}}(b''_k, pk))).\end{aligned}$$
5) MIC1 chooses $r'_0 \in_R \mathcal{M}_{inv}$ and $r'_1 \in_R \mathcal{M}$ and calculates $(\alpha^\star, \boldsymbol{\beta}^\star) \leftarrow \mathsf{BlindESED}(\boldsymbol{C}'', \boldsymbol{C}', r'_0, r'_1, pk)$.
6) MIC1 provides $S^\star = (\alpha^\star, \boldsymbol{\beta}^\star)$ to MC1.
7) MC1 decrypts $S^\star$ using (2) to acquire $s^\star \in \mathcal{M}$ and sends it to MIC1 in the same way as the real execution.
8) Finally, MIC1 computes $s' \leftarrow (s^\star - r'_1)/(r'_0)^2$, and if $0 \le s' < \tau$, it sends $\mathbf{r}' \leftarrow T(= 1)$ to MC1; otherwise, it sends $\mathbf{r}' \leftarrow F(= 0)$ to MC1.

In steps 6 and 8, MC1 receives $S^\star$ and $\mathbf{r}'$, respectively, from MIC1. Now we will examine whether MC1 can distinguish the simulated values of $S^\star$ and $\mathbf{r}'$ from those in the real world. First, it is obvious that MC1 cannot distinguish between $S^\star$ in the real world and $S^\star$ in the ideal world, because in both worlds $S^\star$ is a ciphertext to a random plaintext in $\mathcal{M}$ owing to the randomizer r_1. Therefore, we focus on the distribution of $\mathbf{r}'$. If MC1 honestly follows the protocol, the above MIC1's simulation always produces $\mathbf{r}' = \mathbf{r}$ because MC1 receives $\mathbf{r}$ from TP. This is also true for the case where MC1 manipulates $\boldsymbol{C}'$. On the other hand, if MC1 manipulates $s^\star$, it will affect s in the real world and s' in the ideal world. Let s_c and s'_c denote the modified values of s and s', respectively, when MC1 modifies $s^\star$ to $s^\star + \epsilon$. Then, $s_c = u(s^\star + \epsilon - r_1)$ and $s'_c = v(s^\star + \epsilon - r'_1)$ if we define $u = (r_0^2)^{-1}$ and $v = ((r'_0)^2)^{-1}$ for concise notation. The amounts of change in s and s' are $s_c - s = u\epsilon$ and $s'_c - s' = v\epsilon$, respectively. Because u and v are random, $u\epsilon$ and $v\epsilon$ are also random owing to the modulus wraparounds in the underlying LHE. Thus s_c and s'_c are random. Therefore, the same ϵ may affect randomly the reply $\mathbf{r}$ from the real-world server S and the reply $\mathbf{r}'$ from MIC1. Let $\mathbf{r_c}$ and $\mathbf{r_c}'$ be the changed values of $\mathbf{r}$ and $\mathbf{r}'$, respectively. If $\mathbf{r_c}$ and $\mathbf{r_c}'$ may have different distributions, MC1 will be able to distinguish between the real-world execution and the ideal-world execution with probability up to $\Pr[\mathbf{r_c} \neq \mathbf{r_c}']$. To estimate this probability, observe that $\Pr[\mathbf{r_c} = 1] = \tau/|\mathcal{M}|$ for a random s_c and $\Pr[\mathbf{r_c}' = 1] = \tau/|\mathcal{M}|$ for a random s'_c. Then, $\Pr[\mathbf{r_c} \neq \mathbf{r_c}'] = \Pr[\mathbf{r_c} = 1 \wedge \mathbf{r_c}' = 0] + \Pr[\mathbf{r_c} = 0 \wedge \mathbf{r_c}' = 1] = 2(\tau/|\mathcal{M}|)(1 - (\tau/|\mathcal{M}|)) < 2\tau/|\mathcal{M}|$. In general, $2\tau/|\mathcal{M}|$ is negligible, as explained in Section V-B. Namely, MC1 cannot computationally distinguish between the real-world execution and the ideal-world execution.

Through the above construction, we showed that there exists a PPT algorithm MIC1 that satisfies the condition stated in Theorem 3. This completes the proof. □

Appendix D
Proof of Theorem 4

It is sufficient to prove $\Pr[\mathsf{forge}(\boldsymbol{C}')] + \Pr[\mathsf{forge}(s^\star)] = \mathbf{c} + \tau/|\mathcal{M}|$. First, recall that each element C'_i in $\boldsymbol{C}'$ is in the form $(\Gamma'_i - b, \widehat{\mathsf{Enc}}(b, pk)) \in \mathcal{M} \times \widehat{\mathcal{C}}$, i.e., a plaintext-ciphertext pair of the underlying LHE scheme. Because the LHE scheme of Π is a uniform encryption scheme, every possible plaintext is mapped to the same number of ciphertexts. Therefore, the probability that a randomly chosen $\boldsymbol{C}'$ may satisfy $\mathsf{SED}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}') < \tau$ is exactly the same as the probability that a randomly chosen $\boldsymbol{\Gamma}'$ may satisfy this condition. That is, $\Pr[\mathsf{forge}(\boldsymbol{C}')] = \Pr[\mathsf{SED}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}') < \tau] = \mathbf{c}$. Next, the probability that a randomly chosen s may satisfy $s < \tau$ is $\tau/|\mathcal{M}|$. Because the relation between s and $s^\star$ is a one-to-one mapping, the probability that s computed from a randomly chosen $s^\star$ may satisfy $s < \tau$ is also $\tau/|\mathcal{M}|$. That is, $\Pr[\mathsf{forge}(s^\star)] = \tau/|\mathcal{M}|$. This completes the proof. □

Appendix E
Experimental Proofs

To confirm the security of the proposed system, we also performed experimental proofs against the three possible adversaries: HBCS, MC1, and MC2.

First, to show the security of Π against HBCS, we designed an experiment to verify whether HBCS can actually generate the same set of transcripts as the real protocol Π, without participating in Π. As it is computationally infeasible to generate all possible transcripts, we reduced the parameters

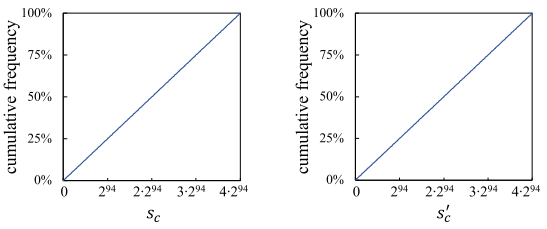

Fig. 9. Cumulative frequency distributions of s_c and s'_c affected by MC1.

to small practical values, without a loss of generality. We set the dimension of the feature vector as 1, and the size of $\mathcal{M}$ as 4 bits. We also set the size of $\widehat{\mathcal{C}}$ as 10 bits. Thus, we used $N = 17^2$ as the modulus of the underlying LHE, which was the only 10-bit value compatible with [20]. For simplicity, we generated the transcripts according to the first case of the proof in Appendix B, i.e., under the assumption that s is always zero for a legitimate user. As a result, we generated $2^4 \times 17^3 = 78,608$ transcripts for each combination (r_0, r_1). Finally, we successfully verified that these transcripts were exactly the same as those obtained by performing the real protocol.

Next, to experimentally prove the security of Π against MC1, we designed another experiment to verify whether there is any recognizable difference between the simulation output of MIC1 and the output of the real server S. Recall that the MIC1 and S outputs are $(S^\star, \mathbf{r})$ and $(S^\star, \mathbf{r}')$, respectively. According to [45], the $\mathcal{HE}'$ ciphertext $S^\star$ has a uniform distribution, so we only examined what happens to the distributions of s and s' associated with $\mathbf{r}$ and $\mathbf{r}'$, respectively, when MC1 has manipulated its input $s^\star$ as in the proof given in Appendix C. To be precise, we analyzed the distributions of s_c and s'_c, which are the values of s and s' affected by MC1's modification of $s^\star$. We simulated this modification 1 million times. Fig. 9 shows the cumulative frequency distributions of s_c and s'_c. We can see that both distributions are almost uniform, i.e., MC1's manipulation randomly affects s_c and s'_c. As a result, we verified that there was no single case in which $\mathbf{r}$ becomes different from $\mathbf{r}'$, among the 1 million trials. In all cases, they were both 0.

Finally, we performed an experimental proof of the security of Π against MC2 considering the following cases:

1) MC2 forges a feature vector $\boldsymbol{\Gamma}'$.
2) MC2 forges a ciphertext vector $\boldsymbol{C}'$.
3) MC2 forges a response $s^\star$.

For the first case, we selected 1 million random feature vectors $\boldsymbol{\Gamma}'$ from the valid range and attempted an authentication with each of these forged feature vectors. However, we did not succeed in any of the authentications. Next, if MC2 wants to be successfully authenticated by forging $\boldsymbol{C}'$, it is natural that MC2 chooses a ciphertext of a valid feature vector, rather than a random ciphertext that may be associated with a plaintext far from the form of a biometric feature vector. Therefore, the success probability of the second case cannot be greater than that of the first case. Finally, we attempted authentication by randomly modifying $s^\star$ 1 million times, but never succeeded in an authentication.

Through the above three experiments, we have verified experimentally that the proposed system is secure against the three possible adversaries. □

REFERENCES

[1] P. Phillips, H. Moon, S. Rizvi, and P. Rauss, "The FERET evaluation methodology for face-recognition algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 10, pp. 1090–1104, Oct. 2000.

[2] Y.-H. Jo, S.-Y. Jeon, J.-H. Im, and M.-K. Lee, "Security analysis and improvement of fingerprint authentication for smartphones," *Mobile Inf. Syst.*, vol. 2016, Mar. 2016 Art. no. 8973828.

[3] N. K. Ratha, J. H. Connell, and R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Syst. J.*, vol. 40, no. 3, pp. 614–634, 2001.

[4] N. K. Ratha, S. Chikkerur, J. H. Connell, and R. M. Bolle, "Generating cancelable fingerprint templates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 561–572, Apr. 2007.

[5] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proc. 6th ACM Conf. Comput. Commun. Secur. (CCS)*, 1999, pp. 28–36.

[6] F. Quan, S. Fei, C. Anni, and Z. Feifei, "Cracking Cancelable Fingerprint Template of Ratha," in *Proc. Int. Symp. Comput. Sci. Comput. Technol.*, 2008, pp. 572–575.

[7] S. W. Shin, M.-K. Lee, D. Moon, and K. Moon, "Dictionary attack on functional transform-based cancelable fingerprint templates," *ETRI J*, vol. 31, no. 5, pp. 628–630, Oct. 2009.

[8] A. Nagar, K. Nandakumar, and A. K. Jain, "Biometric template transformation: A security analysis," *Proc. SPIE*, vol. 7541, Feb. 2010, Art. no. 75410O.

[9] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith, "Secure remote authentication using biometric data," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2005, pp. 147–163.

[10] T. Bhattasali, K. Saeed, N. Chaki, and R. Chaki, "A survey of security and privacy issues for biometrics based remote authentication in cloud," in *Proc. Int. Conf. Comput. Inf. Syst. Ind. Manage.*, 2014, pp. 112–121.

[11] J. Bringer, H. Chabanne, and A. Patey, "Privacy-preserving biometric identification using secure multiparty computation: An overview and recent trends," *IEEE Signal Process. Mag.*, vol. 30, no. 2, pp. 42–52, Mar. 2013.

[12] Z. Rui and Z. Yan, "A survey on biometric authentication: Toward secure and privacy-preserving identification," *IEEE Access*, vol. 7, pp. 5994–6009, 2019.

[13] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Found. Secure Comput.*, vol. 4, no. 11, pp. 169–179, Oct. 1978.

[14] S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, Apr. 1984.

[15] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, vol. 1999, pp. 223–238.

[16] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proc. Theory Cryptogr. Conf.*, 2005, pp. 325–341.

[17] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Symp. Theory Comput. (STOC)*, 2009, pp. 169–178.

[18] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2010, pp. 24–43.

[19] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *TOCTACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 1–36, Jul. 2014.

[20] M. Joye and B. Libert, "Efficient cryptosystems from 2 k-th power residue symbols," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2013, pp. 76–92.

[21] A. C.-C. Yao, "How to generate and exchange secrets," in *Proc. 27th Annu. Symp. Found. Comput. Sci. (SFCS)*, Oct. 1986, pp. 162–167.

[22] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Proc. CRYPTO*, 2012, pp. 643–662.

[23] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, "Practical covertly secure MPC for dishonest majority—Or: Breaking the SPDZ limits," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2013, pp. 1–18.

[24] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," *J. Cryptol.*, vol. 23, no. 2, pp. 281–343, 2010.

[25] Y. Lindell, "Tutorials on the foundations of cryptography," in *How to Simulate It—A Tutorial on the Simulation Proof Technique*. Cham, Switzerland: Springer, 2017, pp. 277–346.

[26] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2009, pp. 235–253.
[27] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *Proc. Int. Conf. Inform. Secur. Cryptol.*, 2009, pp. 229–244.
[28] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich, "SCiFI–a system for secure face identification," in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 239–254..
[29] M. A. Turk and A. P. Pentland, "Face recognition using Eigenfaces," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jan. 1991, pp. 586–591.
[30] M. Barni *et al.*, "Privacy-preserving fingercode authentication," in *Proc. 12th ACM Workshop Multimedia Secur. (MM&Sec)*, 2010, pp. 231–240.
[31] Y. Huang, L. Malka, D. Evans, and J. Katz, "Efficient privacy-preserving biometric identification," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2011, pp. 2652–2660.
[32] M. Blanton and P. Gasti, "Secure and efficient protocols for iris and fingerprint identification," in *Proc. 16th Eur. Conf. Res. Comput. Secur.*, 2011, pp. 190–209.
[33] D. Demmler, T. Schneider, and M. Zohner, "ABY–a framework for efficient mixed-protocol secure two-party computation," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 770–778.
[34] H. Chun, Y. Elmehdwi, F. Li, P. Bhattacharya, and W. Jiang, "Outsourceable two-party privacy-preserving biometric authentication," in *Proc. 9th ACM Symp. Inf., Comput. Commun. security (ASIA CCS)*, 2014, pp. 401–412.
[35] J. H. Cheon, H. Chung, M. Kim, and K.-W. Lee. (2016) *Ghostshell: Secure Biometric Authentication Using Integrity-Based Homomorphic Evaluations*. [Online]. Available: https://eprint.iacr.org/2016/484
[36] J. Im, J. Choi, D. Nyang, and M. Lee, "Privacy-preserving palm print authentication using homomorphic encryption," in *Proc. 2nd Int. Conf. Big Data Intell. Comput.*, 2016, pp. 878–881.
[37] D. Lin, N. Hilbert, C. Storer, W. Jiang, and J. Fan, "UFace: Your universal password that no one can see," *Comput. Secur.*, vol. 77, pp. 627–641, Aug. 2018.
[38] S. F. Shahandashti, R. Safavi-Naini, and N. A. Safa, "Reconciling user privacy and implicit authentication for mobile devices," *Comput. Secur.*, vol. 53, pp. 215–233, Sep. 2015.
[39] J. Sedenka, S. Govindarajan, P. Gasti, and K. S. Balagani, "Secure Outsourced Biometric Authentication With Performance Evaluation on Smartphones," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 2, pp. 384–396, Feb. 2015.
[40] P. Gasti, J. Sedenka, Q. Yang, G. Zhou, and K. S. Balagani, "Secure, fast, and energy-efficient outsourced authentication for smartphones," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2556–2571, Nov. 2016.
[41] A. Abidin, "On privacy-preserving biometric authentication," in *Proc. Inf. Secur. Cryptol.*, 2017, pp. 169–186.
[42] H. Gunasinghe and E. Bertino, "PrivBioMTAuth: Privacy preserving biometrics-based and user centric protocol for user authentication from mobile phones," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 4, pp. 1042–1057, Apr. 2018.
[43] G. Droandi, M. Barni, R. Lazzeretti, and T. Pignata, "SEMBA:SEcure multi-biometric authentication," Mar. 2018, *arXiv:1803.10758*. [Online]. Available: https://arxiv.org/abs/1803.10758
[44] D. Catalano and D. Fiore, "Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2015, pp. 1518–1529.
[45] D. Catalano and D. Fiore. (2014) *Boosting linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data*. [Online]. Available: https://eprint.iacr.org/2014/813
[46] *The Open MP*. Accessed: Dec. 16, 2019. [Online]. Available: https://www.openmp.org/
[47] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1701–1708.
[48] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2879–2886.
[49] T. Hassner, S. Harel, E. Paz, and R. Enbar, "Effective face frontalization in unconstrained images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4295–4304.
[50] D. Chen, X. Cao, F. Wen, and J. Sun, "Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3025–3032.
[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[52] *The GNU MP Bignum Library*. Accessed: Dec. 16, 2019. [Online]. Available: https://gmplib.org/
[53] *Open CV*. Accessed: Dec. 16, 2019. [Online]. Available: https://opencv.org/
[54] *Dlib C++ Library*. Accessed: Dec. 16, 2019. [Online]. Available: http://dlib.net/
[55] *An optimized BLAS*. Accessed: Dec. 16, 2019. [Online]. Available: https://www.openblas.net/
[56] *DLIB-Models*. Accessed: Dec. 16, 2019. [Online]. Available: https://github.com/davisking/dlib-models
[57] *The Database of Faces formerly The ORL Database of Faces*. Accessed: Dec. 16, 2019. [Online]. Available: http://cam-orl.co.uk/facedatabase.html
[58] *Personal Information Protection Act*. Accessed: Dec. 16, 2019. [Online]. Available: http://www.law.go.kr/lsInfoP.do?lsiSeq=142563&viewCls=engLsInfoR
[59] S. Sengupta, J.-C. Chen, C. Castillo, V. M. Patel, R. Chellappa, and D. W. Jacobs, "Frontal to profile face verification in the wild," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2016, pp. 1–9.
[60] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.
[61] J. Yang, D. Zhang, A. Frangi, and J.-y. Yang, "Two-dimensional PCA: A new approach to appearance-based face representation and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 131–137, Jan. 2004.
[62] M. Keller, E. Orsini, and P. Scholl, "MASCOT: Faster malicious arithmetic secure computation with oblivious transfer," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2016, pp. 830–842.
[63] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing, "SPDZ2k: Efficient MPC mod 2k for dishonest majority," in *Proc. CRYPTO*, 2018, pp. 769–798.
[64] *MP-SPDZ*. Accessed: Dec. 16, 2019. [Online]. Available: https://github.com/data61/MP-SPDZ
[65] *Open SSL*. Accessed: Dec. 16, 2019. [Online]. Available: https://www.openssl.org/
[66] *Digital Signature Standard (DSS)*, Standard FIPS PUB 186-4, Nat. Inst. Standards Technol. Std., 2013.

Jong-Hyuk Im received the B.S. and M.S. degrees in computer engineering from Inha University, South Korea, in 2013 and 2015, respectively, where he is currently pursuing the Ph.D. degree. His research interests include applied cryptography, biometrics, and information security.

Seong-Yun Jeon received the B.S. degree in computer engineering from Inha University, South Korea, in 2018, where he is currently pursuing the M.S. degree. His research interests include applied cryptography, biometrics, and information security.

Mun-Kyu Lee (Member, IEEE) received the B.S. and M.S. degrees in computer engineering and the Ph.D. degree in electrical engineering and computer science from Seoul National University, in 1996, 1998, and 2003, respectively. From 2003 to 2005, he was a Senior Engineer with the Electronics and Telecommunications Research Institute, South Korea. He is currently a Professor with the Department of Computer Engineering, Inha University, South Korea. His research interests include cryptographic algorithms, information security, and theory of computation.