Hide-and-Seek: Face Recognition in Private

Yogachandran Rahulamathavan, and Muttukrishnan Rajarajan
Information Security Group, School of Engineering, Mathematics, and Computer Science,
City University London, UK, EC1V 0HB, Emails:{Yogachandran.Rahulamathavan.1,R.Muttukrishnan}@city.ac.uk

***Abstract*—Recent trend towards cloud computing and outsourcing has led to the requirement for face recognition (FR) to be performed remotely by third-party servers. When outsourcing the FR, client's test image and classification result will be revealed to the servers. Within this context, we propose a novel privacy-preserving (PP) FR algorithm based on randomization. Existing PP FR algorithms are based on homomorphic cryptography which requires higher computational power and communication bandwidth. Since we use randomization, the proposed algorithm outperforms the homomorphic algorithm in terms of computational and communication complexity. We validated our algorithm using popular ORL database. Experimental results demonstrate that accuracy of the proposed algorithm is same as the accuracy of existing algorithms, while improving the computational complexity by 120 times and communication complexity by 2.5 times against the existing homomorphic cryptography based approach.**

I. Introduction

Face recognition (FR) is a computer application for automatically identifying a person from a digital image. One of the ways to do this is by comparing selected facial features from the image and a facial database. Building a FR classifier requires a large number of valid training samples; hence, it is infeasible for individuals or small organizations to build their own classifier. One viable solution to this problem is to outsource the FR to a more computationally powerful third-party. Outsourcing the FR mitigates the requirement of not only a large number of valid training data samples but also high computational, storage resources, and maintenance requirements for clients (i.e., individuals or small organizations).

In general, FR problem can be solved using mathematical model which is based on two phases: training phase and testing phase.Traditionally, training and testing were performed within the same environment (i.e., as shown in Fig. 1a, client and server belong to the same party). However, when outsourcing the FR to the third-party (see Fig. 1b), the client and server become mutually untrusted parties and releasing the data samples owned by the client to the server raises privacy concerns [1], [2]. Furthermore, the server may not wish to disclose any details or parameters of its training data set even if it offers classification service to the client. In order to mitigate these privacy vulnerabilities, in literature, privacy-preserving (PP) FR algorithm is proposed based on homomorphic cryptography [7]. However, the existing PP FR algorithm requires higher computational power and communication bandwidth. Homomorphic cryptosystems such as Paillier are public key cryptosystems and uses large keys for encryption and decryption [15]. This involves exponentiation of very large number, which is computationally intensive operation. On the other hand, the homomorphic encryption used for data classification schemes can perform limited number of operations in encrypted domain

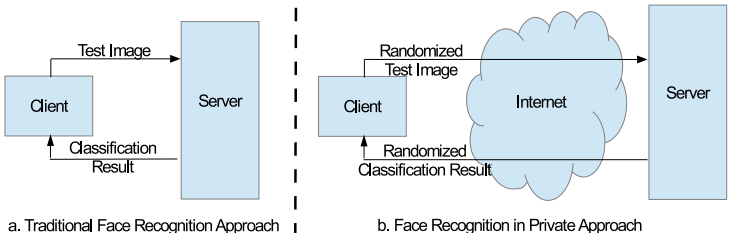

Fig. 1. Comparison between traditional and private face recognition approaches.

(either addition or multiplication). Hence, more number of interactions between client and server is required to obtain the classification result. Hence, in this paper, we propose a lightweight PP FR protocol based on randomization technique. Note that the proposed algorithm can be directly applied for various potential applications such as emotion detection, biometric classification, and clinical decision support.

Notation. We use boldface lower case letters to denote vectors; $(.)'$ denotes the transpose operator; $\|.\|_2$ the Euclidean norm; $\lfloor . \rceil$ the nearest integer approximation; and $sign(m)$ denotes sign of the number m. The modular reduction operator is denoted by mod; $vec()$ is a vectorization operator that forms a vector by stacking the columns of a matrix; $\otimes$ denotes the Kronecker product which is an operation on two matrices of arbitrary size resulting in a block matrix operation.

II. Face Recognition: A Conventional Approach

An efficient approach to recognize human faces is to transforms face images into characteristic feature vectors of a lower dimensional vector space [3]. Principal component analysis (PCA) is a well-known feature extraction method which aims to obtain a set of mutually orthogonal bases that describe the global information of the data points in terms of variance [3]. In this paper, we consider only PCA based feature extraction, however, the proposed algorithm can be directly extended for more sophisticated feature extraction methods such as Fisher linear discriminant analysis and variants of Fisher linear discriminant analysis.

A. Feature Extraction

Suppose we have C number of facial image classes. Let n_c be the number of facial images for cth class. The total number of facial images can thus be denoted as $N = \sum_{c=1}^{C} n_c$. Let us assume that each facial image is a real valued grayscale image and can be represented as a matrix, where each element corresponds to the pixel value of a point within the image. A two-dimensional facial image matrix can be converted to a

one-dimensional vector by stacking each column (or row) of the matrix into a long vector. Denote $\widetilde{\mathbf{x}_i} \in \mathbb{Z}^{n\times 1}$ as a vector representation of the ith facial image.

Let us start with a training set of facial images $\widetilde{\mathbf{x}_i} = [\widetilde{x_{1,i}}, \ldots, \widetilde{x_{n,i}}]^T \in \mathbb{Z}^{n\times 1}$, $i = 1, \ldots, N$. Using these training data samples, we can train a classifier to classify an unlabeled test sample. Initially, the training data needs to be normalized to keep the numeric values of training samples on the same scale. Let us denote the normalized training data samples as $\mathbf{x}_i \in \mathbb{R}^{n\times 1}$, $i = 1, \ldots, N$ where,

$$\mathbf{x}_i = \tilde{\mathbf{x}}_i - \bar{\mathbf{x}}, \forall i, \tag{1}$$

where $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \tilde{\mathbf{x}}_i$, denotes the mean of the training data samples. Let us define a matrix $\mathbf{X}$ as follows: $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2, \ldots, \mathbf{x}_N]$. Now the covariance matrix, $\mathbf{C}$, of the normalized training data samples can be defined as follows:

$$\mathbf{C} = \frac{1}{N}\mathbf{X}\mathbf{X}^T. \tag{2}$$

The Eigenfaces are the eigenvectors of the covariance matrix or scatter matrix, $\mathbf{C}$, defined in (2). The transformation matrix $\mathbf{U}$ can be obtained using the following optimization

$$\mathbf{U} = \underset{\mathbf{V}}{argmax} \mid \mathbf{V}^T\mathbf{C}\mathbf{V} \mid = [\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_m], \tag{3}$$

where $\{\mathbf{u}_i \in \mathbb{R}^{n\times 1} \mid i = 1, \ldots m\}$ is a set of eigenvectors corresponding to the m largest eigenvalues of $\mathbf{C}$ (since the size of $\mathbf{C}$ makes it computationally infeasible to directly compute eigenvectors, usually smaller matrix $\mathbf{X}^T\mathbf{X}$ and appropriate post-processing are being used).

Let us denote the projected image in the lower dimensional feature space corresponding to the image $\mathbf{x}_i = [x_{1,i}, \ldots, x_{n,i}]^T \in \mathbb{R}^{n\times 1}$ as $\mathbf{y}_i = [y_{1,i}, \ldots, y_{m,i}]^T \in \mathbb{R}^{m\times 1}$, where $m \ll n$. Hence, $\mathbf{y}_i$ can be obtained by the following linear projection:

$$\mathbf{y}_i = \mathbf{U}^T\mathbf{x}_i,\ i = 1, \ldots, N, \tag{4}$$

where

$$y_{k,i} = \mathbf{u}_k^T\mathbf{x}_i,\ k = 1, \ldots, m. \tag{5}$$

B. Classification

After feature extraction, the classification can be based on 1-nearest-neighbor classifier (1-NN) [4]. 1-NN classifier computes the matching class of the test image based on the closest training image. Squared Euclidean distance calculation can be used to obtain the distances between the test image and training images (i.e., in the lower dimension). In order to explain the classification phase, let us denote a test image before normalization as $\tilde{\mathbf{t}} = [\widetilde{t_1}, \ldots, \widetilde{t_n}]^T \in \mathbb{Z}^{n\times 1}$ and after normalization as $\mathbf{t} = [t_1, \ldots, t_n]^T \in \mathbb{R}^{n\times 1}$ where

$$\mathbf{t} = \tilde{\mathbf{t}} - \bar{\mathbf{x}}. \tag{6}$$

Denote the low dimensional vector corresponding to $\mathbf{t}$ as $\mathbf{w} = [w_1, \ldots, w_m]^T \in \mathbb{R}^{m\times 1}$ where $m \ll n$. Now $\mathbf{t}$ is projected by the projection matrix as

$$\mathbf{w} = \mathbf{U}^T\mathbf{t}, \tag{7}$$

where

$$w_k = \mathbf{u}_k^T\mathbf{t},\ k = 1, \ldots, m. \tag{8}$$

The squared Euclidean distance, d_i, between $\mathbf{w}$ and $\mathbf{y}_i$, $i = 1, \ldots, N$ is

$$\begin{aligned} d_i &= \|(\mathbf{w} - \mathbf{y}_i)\|_2^2 = \|(\mathbf{U}^T\mathbf{t} - \mathbf{U}^T\mathbf{x}_i)\|_2^2, \\ &= \sum_{k=1}^{m} \left(\mathbf{u}_k^T\mathbf{t}\mathbf{t}^T\mathbf{u}_k - 2\mathbf{u}_k^T\mathbf{t}\mathbf{u}_k^T\mathbf{x}_i + \mathbf{u}_k^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{u}_k\right). \end{aligned} \tag{9}$$

The decision rule of the 1-NN classifier is that the training image $\mathbf{x}_*$ is said to have same features of the test image if

$$d_* = \min\{d_1, \ldots, d_N\}, \tag{10}$$

where d_* is smaller than a given threshold T.

III. Face recognition: In private

In this section, we show how the traditional algorithm can be extended to work in private, so that any such test image can be recognized with the aid of a third-party server but without leaking any of the image contents or the recognition results to the server. The new algorithm satisfies the following three requirements: R_1- without public-key homomorphic encryption schemes, R_2- hide the client's data and the result from the server, and R_3- hide the server side parameters from the client. In the following subsections, we explain how the steps involved in the conventional approach can be computed without violating the above three requirements (R_1 – R_3).

A. Normalizing the test image in private

Initially, client's test image needs to be normalized before the classification. However, the client cannot send the test image, $\tilde{\mathbf{t}}$, due to the privacy concerns. Client can add noise to the test image before sending it to the server; however, since the facial image has a unique structure, adding noise may not effectively hide the test image. Hence, the client only sends a noise vector, $\tilde{\boldsymbol{\eta}} \in \mathbb{Z}^{n\times 1}$, with the same dimension as the test image. Since server receives only the noise vector, he cannot be able get any information about the test image vector. As shown in (6), server normalizes the noise vector, $\tilde{\boldsymbol{\eta}}$, and obtains the normalized noise vector, $\boldsymbol{\eta}$, as follows:

$$\boldsymbol{\eta} = \tilde{\boldsymbol{\eta}} - \bar{\mathbf{x}}. \tag{11}$$

However, only the client knows the difference between the test image and the noise image. Let us denote the difference as $\boldsymbol{\gamma} = \tilde{\boldsymbol{\eta}} - \tilde{\mathbf{t}} \in \mathbb{Z}^{n\times 1}$. Hence, only the client can recover the normalized test image, $\mathbf{t} \in \mathbb{Z}^{n\times 1}$ (i.e., $\mathbf{t} = \tilde{\mathbf{t}} - \bar{\mathbf{x}}$), from the normalized noise vector as follows:

$$\mathbf{t} = \boldsymbol{\eta} - \boldsymbol{\gamma}. \tag{12}$$

B. Projecting the normalized noise onto the lower dimension space in private

Since the server has obtained only the normalized noise vector, he can only project the normalized noise vector onto the lower dimension space instead of normalized test image. Hence, as shown in (7), the server projects the normalized noise vector and obtains lower dimension vector, $\bar{\mathbf{w}} = [\bar{w_1}, \bar{w_2}, \ldots, \bar{w_m}] \in \mathbb{R}^{m\times 1}$, as follows:

$$\bar{\mathbf{w}} = \mathbf{U}^T\boldsymbol{\eta}, \tag{13}$$

where

$$\bar{w_k} = \mathbf{u}_k^T \boldsymbol{\eta},\ k = 1, \ldots, m. \tag{14}$$

However, using (11), $\boldsymbol{\gamma} = \tilde{\boldsymbol{\eta}} - \tilde{\mathbf{t}}$, $\mathbf{t} = \tilde{\mathbf{t}} - \bar{\mathbf{x}}$, (8) and (14), we can derive $\bar{w_k}$ in terms of w_k, $k = 1, \ldots, m$ as follows:

$$\bar{w_k} = \mathbf{u}_k^T(\tilde{\boldsymbol{\eta}} - \bar{\mathbf{x}}) = \mathbf{u}_k^T[(\tilde{\mathbf{t}} + \boldsymbol{\gamma}) - \bar{\mathbf{x}}] = w_k + \mathbf{u}_k^T\boldsymbol{\gamma},\ \forall k. \tag{15}$$

The scalar $\mathbf{u}_k^T\boldsymbol{\gamma}$ in (15) is unknown to the server. Hence, the server uses only the $\bar{w_k},\ k = 1, \ldots, m$ obtained in (14) for the distance calculation step instead of $w_k,\ k = 1, \ldots, m$.

C. Euclidean distance calculation in private

Since the server has only computed $\bar{\mathbf{w}}$, lets us denote the Euclidean distance between $\bar{\mathbf{w}}$ and the low dimensional training image $\mathbf{y}_i$ as $\bar{d_i},\ i = 1, \ldots, N$. Similar to (9), we can compute the Euclidean distances $\bar{d_i},\ i = 1, \ldots, N$ as

$$\begin{aligned} \bar{d_i} &= \|(\bar{\mathbf{w}} - \mathbf{y}_i)\|_2^2 \ = \ \|(\mathbf{U}^T\boldsymbol{\eta} - \mathbf{U}^T\mathbf{x}_i)\|_2^2, \\ &= \sum_{k=1}^{m} \left(\mathbf{u}_k^T\boldsymbol{\eta}\boldsymbol{\eta}^T\mathbf{u}_k - 2\mathbf{u}_k^T\boldsymbol{\eta}\mathbf{u}_k^T\mathbf{x}_i + \mathbf{u}_k^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{u}_k\right). \end{aligned} \tag{16}$$

Since, $\boldsymbol{\eta} = \mathbf{t} + \boldsymbol{\gamma}$, we can write $\bar{d_i}$ in terms of $d_i,\ i = 1, \ldots, N$ as

$$\begin{aligned} \bar{d_i} &= \sum_{k=1}^{m} \big[\mathbf{u}_k^T(\mathbf{t}+\boldsymbol{\gamma})(\mathbf{t}+\boldsymbol{\gamma})^T\mathbf{u}_k - 2\mathbf{u}_k^T(\mathbf{t}+\boldsymbol{\gamma})\mathbf{u}_k^T\mathbf{x}_i \\ &\qquad + \mathbf{u}_k^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{u}_k\big], \\ &= d_i + \sum_{k=1}^{m} \left[\mathbf{u}_k^T\left(2\mathbf{t}\boldsymbol{\gamma}^T + \boldsymbol{\gamma}\boldsymbol{\gamma}^T - 2\boldsymbol{\gamma}\mathbf{x}_i^T\right)\mathbf{u}_k\right]. \end{aligned} \tag{17}$$

In the next subsection, we elaborate how the distances obtained in (17) can be used to obtain the matching training image.

D. Minimum distance calculation to match the test image to a known class in private

Let us call the difference between d_i and $\bar{d_i}$ in (17) as masking factor and denote it as $m_i,\ i = 1, \ldots, N$ (i.e., $m_i = \sum_{k=1}^{m}\left[\mathbf{u}_k^T(2\mathbf{t}\boldsymbol{\gamma}^T + \boldsymbol{\gamma}\boldsymbol{\gamma}^T - 2\boldsymbol{\gamma}\mathbf{x}_i^T)\mathbf{u}_k\right]$). Let us rewrite (17) as follows:

$$\bar{d_i} = d_i + m_i,\ i = 1, \ldots, N. \tag{18}$$

In order to find the minimum distance, the server needs to remove the masking factor, m_i, from $\bar{d_i}$. Since the server does not know the difference vector, $\boldsymbol{\gamma}$, or the normalized test image $\mathbf{t}$, it is infeasible for the server to obtain a true distance values d_i from $\bar{d_i},\ i = 1, \ldots, N$. Hence, the server cannot find the matching training image corresponding to the test image using (18). In order to obtain the matching image, the server needs to interact with the client by sending all the $\bar{d_i},\ i = 1, \ldots, N$. Before sending $\bar{d_i},\ i = 1, \ldots, N$, the server generates random value r_i for each $\bar{d_i}$ and send $\widetilde{d_i}$ to the client where

$$\widetilde{d_i} = \bar{d_i} + r_i = d_i + m_i + r_i,\ i = 1, \ldots, N. \tag{19}$$

Now the client must interact with the server to compute $m_i + r_i,\ i = 1, \ldots, N$. Using $m_i + r_i,\ i = 1, \ldots, N$, the client can compute the actual Euclidean distances $d_i,\ i = 1, \ldots, N$ as follows:

$$d_i = \widetilde{d_i} - (m_i + r_i),\ i = 1, \ldots, N, \tag{20}$$

$$m_i + r_i = \sum_{k=1}^{m} \mathbf{u}_k^T(2\mathbf{t}\boldsymbol{\gamma}^T + \boldsymbol{\gamma}\boldsymbol{\gamma}^T - 2\boldsymbol{\gamma}\mathbf{x}_i^T)\mathbf{u}_k + r_i.$$

TABLE I
PRIVACY-PRESERVING SECURE TWO-PARTY COMPUTATION ALGORITHM TO COMPUTE $r + \mathbf{a}^T\mathbf{b}$ PRIVATELY.

1. **Input by Client:** $\mathbf{a} = [a_1, \ldots, a_l]^T$, where $l < 2^{16}$, $\{(a_i)_{\forall\ i}\} \in \mathbb{Z}_{o_1}$ with $o_1 \leq 2^8$
Input by Server: r, $\lfloor s.\mathbf{b} \rceil = [\lfloor s.b_1 \rceil, \ldots, \lfloor s.b_l \rceil]^T$, where $l < 2^{16}$, $\{s.r,\ (b_i)_{\forall\ i}\} \in \mathbb{Z}_{o_2}$ with $o_2 \leq 2^{32}$
2. **Output:** $r + \frac{1}{s}(\lfloor s\mathbf{a} \rceil^T\mathbf{b})$ (known only to the client)
Client first performs the following operations
3. choose primes α, β, where $\lvert\alpha\rvert = 256$ bits and $\beta > (l.o_1.o_2 + o_2 + 1).\alpha^2$, e.g., $\lvert\beta\rvert > 312$ bits if $l = 2^{16}$
4. set $K = 0$ and choose l positive random integers $(c_1, \ldots, c_l)$ such that $o_2.\sum_{i=1}^{n} c_i < \alpha - o_2.l$
5. **for** each element $b_i \in \mathbf{b}$ **do**
6. choose random integer z_i, compute $z_i.\beta$ such that $\lvert z_i.\beta\rvert \approx 1024$ bits, and calculate $k_i = z_i.\beta - c_i$
7. **if** $b_i > 0$ **then**
8. $C_i = a_i.\alpha + c_i + z_i.\beta$, $K = K + k_i$
9. **else if** $b_i = 0$ **then**
10. $C_i = c_i + z_i.\beta$, $K = K + k_i$
11. **end if**
12. **end for**
13. send $(\alpha, C_1, C_2, \ldots, C_n)$ to the server
Now server execute the following operations
14. **for** each element $b_i \in \mathbf{b}$ **do**
15. **if** $\lfloor s.b_i \rceil > 0$ **then**
16. $D_i = \lfloor s.b_i \rceil.\alpha.C_i$ i.e., if $a_i > 0$ $\rightarrow D_i = \lfloor s.b_i \rceil.a_i.\alpha^2 + \lfloor s.b_i \rceil.\alpha.c_i + \lfloor s.b_i \rceil.\alpha.z_i.\beta$ if $a_i = 0 \rightarrow D_i = \lfloor s.b_i \rceil.\alpha.c_i + \lfloor s.a_i \rceil.\alpha.z_i.\beta$
17. **else if** $b_i = 0$ **then**
18. $D_i = C_i$ i.e., if $a_i > 0 \rightarrow D_i = a_i.\alpha + c_i + z_i.\beta$ if $a_i = 0 \rightarrow D_i = c_i + z_i.\beta$
19. **end if**
20. **end for**
21. compute $D = s.r.\alpha^2 + \sum_{i=1}^{l} D_i$ and return D to the client
Now client continues to perform the following operations
22. compute $E = D + K\ mod\ \beta$
23. **return** $\frac{E - (E\ mod\ \alpha^2)}{s.\alpha^2}$ which is equal to $r + \frac{1}{s}(\lfloor s\mathbf{b} \rceil^T\mathbf{a}) \approx \mathbf{a}^T\mathbf{b} + r$
24. **end procedure**

Since $\mathbf{t} = \tilde{\mathbf{t}} - \bar{\mathbf{x}}$,

$$\begin{aligned} m_i + r_i &= \sum_{k=1}^{m} \mathbf{u}_k^T[2(\tilde{\mathbf{t}} - \bar{\mathbf{x}})\boldsymbol{\gamma}^T + \boldsymbol{\gamma}\boldsymbol{\gamma}^T - 2\boldsymbol{\gamma}\mathbf{x}_i^T]\mathbf{u}_k + r_i, \\ &= \sum_{k=1}^{m} \mathbf{u}_k^T[(2\tilde{\mathbf{t}} + \boldsymbol{\gamma})\boldsymbol{\gamma}^T]\mathbf{u}_k - \sum_{k=1}^{m} \mathbf{u}_k^T(2\bar{\mathbf{x}}\boldsymbol{\gamma}^T + 2\boldsymbol{\gamma}\mathbf{x}_i^T)\mathbf{u}_k + r_i. \end{aligned} \tag{21}$$

It should be noted that masking values (i.e., $m_i + r_i,\ i = 1, \ldots, N$) should only be known to the client. If the masking values are known to the server, then the server can compute the actual Euclidean distances between the test and training images, and eventually the server can obtain the matching image corresponding to the test image which violates the client's privacy. In (21), the vectors $\tilde{\mathbf{t}}$, and $\boldsymbol{\gamma}$ are only known to the client and the vectors $\mathbf{u}_k$, $\mathbf{x}_i$, $\bar{\mathbf{x}}$ and the random scalar r_i are known only to the server. The vectors and scalars known to the server and client are mixed with each other in (21) and it is difficult to develop PP algorithm to securely compute the masking values $m_i + r_i,\ \forall i$ at present. Hence, we need to reorder (21) such that, the variables known to the server to be on one side while the variables known to the client to other side. In order to compute $(m_i + r_i)$ in private let us propose a PP secure two-party algorithm in the next subsection.

1) Privacy-preserving secure two-party computation: Let us assume that the client knows a vector $\mathbf{a}$ and the server knows a vector $\mathbf{b}$ and scalar r, and both want to interact with each other in order to compute $\mathbf{a}^T\mathbf{b}+r$, where the outcome $\mathbf{a}^T\mathbf{b}+r$ must be known only to the client. In order to compute $r+\mathbf{a}^T\mathbf{b}$ in private, we use the building block used in [5] for PP scalar multiplication. In order to avoid information leakage due to the floating points during the two-party computation, let us scale the elements in vector $\mathbf{b}$ by a large scalar and approximate into nearest integer. This is a valid operation since we scale the vector, $\mathbf{b}$, and r by a large scalar, s, before computation (i.e., $s.r+\mathbf{a}^T\lfloor s\mathbf{b}\rceil$) and divide the outcome by the same scaling factor s (i.e., $\frac{1}{s}(s.r+\mathbf{a}^T\lfloor s\mathbf{b}\rceil)\approx r+\mathbf{a}^T\mathbf{b}$). Approximation errors due to the nearest integer approximation operation, $\lfloor . \rceil$ is discussed in the Section V.

Table I shows the secure two-party computation algorithm to compute $r+\frac{1}{s}(\lfloor s\mathbf{a}\rceil^T\mathbf{b})$ privately. The security of the algorithm relies on masking the variables using large random noise. Let us explain the algorithm in Table I. The server knows r and $\mathbf{b}$ while client knows $\mathbf{a}$. Let us assume that the client's input $\mathbf{a}$ is composed of integers while server's input $\mathbf{b}$ is composed of floating points. Since the elements of $\mathbf{a}$ are integer, the server converts the elements in $\mathbf{b}$ by scaling and nearest integer approximation operations. This will prevent any information leakage via floating points about the server side classification parameters to the client. Initially, the client add different random noises to each of the elements in $\mathbf{a}$ as shown in Steps 7 to 10 in Table I. Since noise is added to the elements of $\mathbf{a}$, the server cannot be able filter the noise to recover the client side parameters, which satisfies the clients privacy requirement. Now the server execute the Steps from 14 to 21 and return the outcome to the client. In Step 21, the server adds random value $s.r$ to $\sum_{i=1}^{l} D_i$ which protects the server side parameters from clients selective input attack according to the Lemma 1 provided below.

Lemma 1: The algorithm in in Table I is not vulnerable to selective input attack i.e., the client cannot learn the server side parameters by selectively input vector.

Proof: See the Appendix I. ■

Finally, the client does the operations in Steps 22 and 23 to obtain $r_i+\frac{1}{s}(\lfloor s\mathbf{a}\rceil^T\mathbf{b})$. The correctness of the algorithm is provided in the following Lemma.

Lemma 2: There is no overflow errors in the algorithm in Table I due to the modulo reduction.

Proof: See the Appendix II. ■

2) Solve (21) Using the Algorithm in Table I: In this section, we show how we can compute (m_i+r_i) in (21) using the algorithm in Table I. In order to exploit the algorithm in Table I, we must rearrange the variables in (21). In order to do this, let us introduce the following Kronecker identities [17]: $vec(\mathbf{AXB})=(\mathbf{B}^T\otimes\mathbf{A})vec(\mathbf{X})$, and $(\mathbf{A}\otimes\mathbf{B})(\mathbf{C}\otimes\mathbf{D})=(\mathbf{AC})\otimes(\mathbf{BD})$, where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$, and $\mathbf{X}$ are matrices/vectors with appropriate sizes. Let us now process the first term in (21) (i.e., $\sum_{k=1}^{m}\mathbf{u}_k^T[(2\tilde{\mathbf{t}}+\boldsymbol{\gamma})\boldsymbol{\gamma}^T]\mathbf{u}_k$) as follows:

$$\begin{aligned}\mathbf{u}_k^T[(2\tilde{\mathbf{t}}+\boldsymbol{\gamma})\boldsymbol{\gamma}^T]\mathbf{u}_k &= vec\left\{\mathbf{u}_k^T[(2\tilde{\mathbf{t}}+\boldsymbol{\gamma})\boldsymbol{\gamma}^T]\mathbf{u}_k\right\},\\ &=(\mathbf{u}_k^T\otimes\mathbf{u}_k^T)[\boldsymbol{\gamma}\otimes(2\tilde{\mathbf{t}}+\boldsymbol{\gamma})]=(\mathbf{u}_k^T\boldsymbol{\gamma})\otimes\left[\mathbf{u}_k^T(2\tilde{\mathbf{t}}+\boldsymbol{\gamma})\right],\\ &=(\mathbf{u}_k^T\boldsymbol{\gamma})\times\left[\mathbf{u}_k^T(2\tilde{\mathbf{t}}+\boldsymbol{\gamma})\right]. \qquad (22)\end{aligned}$$

Let us define the following random variables $r_{i,1}^k$, $r_{i,2}^k$, $r_{i,3}^k$, and $r_{i,4}^k$, $\forall k$ and assume that these random variables are generated by the server to randomize the scalar product output. If we incorporate these random variables within (22) then (22) become equals to (23) which is shown in the top of the next page.

In (23), the server knows $\mathbf{u}_k$, $r_{i,1}^k$, $r_{i,2}^k$, $r_{i,3}^k$, and $r_{i,4}^k$ while the client knows $\boldsymbol{\gamma}$ and $2\tilde{\mathbf{t}}+\boldsymbol{\gamma}$. Hence, client and server collaboratively compute $\left[\mathbf{u}_k^T\boldsymbol{\gamma}+r_{i,1}^k\right]$,$\left[\mathbf{u}_k^T(2\tilde{\mathbf{t}}+\boldsymbol{\gamma})+r_{i,2}^k\right]$, $\left[\left(r_{i,2}^k\mathbf{u}_k\right)^T\boldsymbol{\gamma}+r_{i,3}^k\right]$, and $\left[\left(r_{i,1}^k\mathbf{u}_k\right)^T(2\tilde{\mathbf{t}}+\boldsymbol{\gamma})+r_{i,4}^k\right]$ using the algorithm in Table I. In order to balance the equation, the server subtract the remaining random value $\left(r_{i,1}^k r_{i,2}^k - r_{i,3}^k - r_{i,4}^k\right)$ in (23) in the next computation. Now let us process the remaining terms in (21) (i.e., $-\sum_{k=1}^{m}\mathbf{u}_k^T(2\bar{\mathbf{x}}\boldsymbol{\gamma}^T+2\boldsymbol{\gamma}\mathbf{x}_i^T)\mathbf{u}_k - r_i$). Assume $r_i = -\sum_{k=1}^{m}(r_{i,1}^k r_{i,2}^k - r_{i,3}^k - r_{i,4}^k)$, hence $-\sum_{k=1}^{m}\mathbf{u}_k^T(2\bar{\mathbf{x}}\boldsymbol{\gamma}^T+2\boldsymbol{\gamma}\mathbf{x}_i^T)\mathbf{u}_k - r_i$ can be transformed into (24) which is shown in the top of the page after next page.

In (24), the server knows $2\left[(\mathbf{u}_k^T\otimes\mathbf{u}_k^T\bar{\mathbf{x}})+(\mathbf{u}_k^T\mathbf{x}_i\otimes\mathbf{u}_k^T)\right]$ and $(r_{i,1}^k r_{i,2}^k - r_{i,3}^k - r_{i,4}^k)$ and the clients knows $\boldsymbol{\gamma}$. Hence, the algorithm in Table I can be exploited by both the client and server in order to compute the $(-\sum_{k=1}^{m}\mathbf{u}_k^T(2\bar{\mathbf{x}}\boldsymbol{\gamma}^T+2\boldsymbol{\gamma}\mathbf{x}_i^T)\mathbf{u}_k - r_i)$. Hence, using (23), (24), and Table I, the client can obtain (m_i+r_i) $\forall i$. This will enable the client to compute the true Euclidean distances between the test image and the training images using (20). Hence, the client can find the smallest Euclidean distance using (10). Only task now left is to find the identity of the training image corresponding to the smallest Euclidean distance. This task must be achieved without leaking the index of the training image corresponding to the smallest Euclidean distance to the server.

3) Privacy-preserving Identity Finding: Let us define a binary vector $\mathbf{d}_b \in \{0,1\}^{N,1}$. If nth Euclidean distance is the smallest distance then, the client generates a binary vector $\mathbf{d}_b$ by setting nth element to 1 while setting all other elements to 0. Let us denote the identity of nth training image as id_n and define another vector called identity vector as $\mathbf{d}_{id}=[id_1,\ id_2\ ,\ldots,id_N]^T$. Client must keep the binary vector $\mathbf{d}_b$ away from the server in order to protect the privacy of the test image. However, the client could exploit the algorithm in Table I to obtain the identity of the test image without revealing the $\mathbf{d}_b$ as follows: The identity of the training image which is corresponding to the smallest Euclidean distance (i.e., lets assume nth training image) could be obtained by computing $\mathbf{d}_b^T\mathbf{d}_{id}$ (i.e., $[0\ 0\ \ldots 0\ 1\ 0\ \ldots\ 0].[id_1\ id_2\ \ldots,id_n,id_{n+1},\ldots,id_N]^T = id_n$). It should be noted that the algorithm in Table I but with different inputs can be used to obtain the correct identity of the matching training image. If the client feeds the binary vector $\mathbf{d}_b$ instead of $\mathbf{a}$ and if the server feeds the identity vector $\mathbf{d}_{id}$ instead of $s.\mathbf{b}$ and 0 for r to the algorithm in Table I, then the output of the algorithm should be equivalent to the identity of the matching training image.

IV. Security and Privacy Analysis

The goal of our protocol is to keep the client's test image and the classification result away from the server and the server's parameters away from the client. Initially, the client was just sending the noise vector $\tilde{\boldsymbol{\eta}} \in \mathbb{Z}^{n\times 1}$ to the server

$$(\mathbf{u}_k^T\boldsymbol{\gamma})\times\Big[\mathbf{u}_k^T(2\tilde{\mathbf{t}}+\boldsymbol{\gamma})\Big] = \Big[\mathbf{u}_k^T\boldsymbol{\gamma}+r_{i,1}^k\Big]\times\Big[\mathbf{u}_k^T(2\tilde{\mathbf{t}}+\boldsymbol{\gamma})+r_{i,2}^k\Big] - \Big[\Big(r_{i,2}^k\mathbf{u}_k\Big)^T\boldsymbol{\gamma}+r_{i,3}^k\Big] - \Big[\Big(r_{i,1}^k\mathbf{u}_k\Big)^T(2\tilde{\mathbf{t}}+\boldsymbol{\gamma})+r_{i,4}^k\Big] - \Big(r_{i,1}^k r_{i,2}^k - r_{i,3}^k - r_{i,4}^k\Big). \quad (23)$$

$$\begin{aligned} -\mathbf{u}_k^T(2\bar{\mathbf{x}}\boldsymbol{\gamma}^T+2\boldsymbol{\gamma}\mathbf{x}_i^T)\mathbf{u}_k - (r_{i,1}^k r_{i,2}^k - r_{i,3}^k - r_{i,4}^k) &= -2vec(\mathbf{u}_k^T\bar{\mathbf{x}}\boldsymbol{\gamma}^T\mathbf{u}_k) - 2vec(\mathbf{u}_k^T\boldsymbol{\gamma}\mathbf{x}_i^T\mathbf{u}_k) - (r_{i,1}^k r_{i,2}^k - r_{i,3}^k - r_{i,4}^k), \\ &= -2\Big[(\mathbf{u}_k^T\otimes\mathbf{u}_k^T\bar{\mathbf{x}}) + (\mathbf{u}_k^T\mathbf{x}_i\otimes\mathbf{u}_k^T)\Big]\, vec(\boldsymbol{\gamma}) - (r_{i,1}^k r_{i,2}^k - r_{i,3}^k - r_{i,4}^k). \quad (24) \end{aligned}$$

Fig. 2. Example facial images from the ORL database.

instead of the true test image $\tilde{\mathbf{t}} \in \mathbb{Z}^{n\times 1}$. From this input, the server can only infer the size of the test image. However, facial features of the test image cannot be inferred from the size of the image, hence, there is no privacy or security issue.

There is no interaction between the server and the client until the minimum distance calculation (Subsection III-D), hence, there is no information leakage in Subsection III-B and Subsection III-C to the client. However, the server may use different $\mathbf{u}_k$ vectors to infer w_k from $\bar{w_k}$ or d_i from $\bar{d_i}$. However, since both w_k and $\mathbf{u}_k^T\boldsymbol{\gamma}$ as well as d_i and $\sum_{k=1}^m \left[\mathbf{u}_k^T\left(2\mathbf{t}\boldsymbol{\gamma}^T+\boldsymbol{\gamma}\boldsymbol{\gamma}^T-2\boldsymbol{\gamma}\mathbf{x}_i^T\right)\mathbf{u}_k\right]$ are depend on $\mathbf{u}_k$, the server cannot be able to infer any client side parameter.

In Subsection III-D, the client and server interact with each other using an algorithm in Table I in order to compute the true Euclidean distances. The output of the algorithm in Table I is known only to the client, hence it is crucial to evaluate whether any malicious client can infer the server side parameters using the output. However, we proved in Lemma 1 that, since the server adds a random value r_i at Step 21 in Table 1, the client cannot learn the server side parameters, which satisfies the privacy requirements of the server.

Let us now analyse the security properties of the algorithm in Table I. Since the client hides the test image within input vector $\mathbf{b}$, malicious server may try to learn elements of $\mathbf{b}$ during the execution of the algorithm in Table I. Hence, the algorithm in Table I must be secure enough to protect elements of $\mathbf{b}$ from the server. If we carefully look at the Steps 8 and 10 in the algorithm in Table I, the elements of $\mathbf{b}$ are randomized by very large positive random integers $c_i + z_i.\beta$, where c_i, z_i, and β are only known to the client. Hence, it is infeasible for the server to extract elements of $\mathbf{b}$ from random noise, hence, the algorithm in Table I is secure enough to protect the client side parameters.

V. Numerical Analysis

In this section, we evaluate the proposed algorithm using the ORL Database of Faces [6]. The ORL database contains 10 facial images of 40 distinct subjects i.e., number of classes equal to 40, totalling 400 images (see Fig. 2). The size of each image is 92×112 pixels with 256 grey levels per pixel i.e., $n = 92 \times 112 = 10304$.

TABLE II
Comparison of computational cost of the proposed scheme against [7] (the size of the images, total number of training images, and the total number of feature vectors are denoted as n, N, and m).

	for	Erkin [7]	Proposed
III-A	Client	$n(C_m + C_e)$	–
	Server	$n(2C_m + C_e)$	–
III-B	Client	–	–
	Server	$m(n-1)C_m + mnC_e$	–
III-C	Client	NC_m	$2nC_m$
	Server	$[(2+5m)C_m + (1+4m)C_e]N$	$mN(n+1)C_m$
IV-D	Client	$(2C_m + C_e)log_2 N$	NC_m
	Server	$(9C_m + 3C_e)log_2 N$	$(n+1)C_m$
Total	Client	$(n + N + 2log_2 N)C_m + (n + log_2 N)C_e$	$(2n+N)C_m$
	Server	$[2n + (n-1)m + (2 + 5m)N + 9log_2 N]C_m + (n + mn + N + 4mN + 3log_2 N)C_e$	$(mN + 1)(n+1)C_m$

The aim of this experimental section is to show that the proposed method achieves the same recognition accuracy as that of the Paillier cryptography based PP FR approach of [7]. Since the approach in [7] does not degrade the performance of the plain domain algorithm, our approach therefore does not degrade the accuracy of the plain domain approach. In order to evaluate the performance, we use 5-fold cross validation for the experiments such that for each subject we use 8 images in the training phase and 2 images for testing phase, hence $N = 8 \times 40 = 320$. We consider the first twelve dominant eigenfaces for projection since there is no improvement in the accuracy i.e., $m = 12$. We tested our algorithm on a computer with a 3.40 GHz Intel(R) Xeon(R) processor and 8GB of RAM running on Windows $64-$operating system. The size of the Paillier security parameter was 1048 bits long. Our proposed private FR is implemented in C++ using GNU GMP library version $4.2.4$. Both the server and client were modelled as different threads of a single program, which pass the variables to each other.

To illustrate the effect of the scalar s in Table I, we have plotted the classification accuracy of the proposed algorithm for five different scalar values in Fig. 3. We also plotted the accuracy of [7] against the scaling factor in Fig. 3. Our algorithm performs better than [7] for the smaller scaling factors. Also our algorithm achieves the maximum accuracy faster than [7]. The crucial point is that the classification accuracies of the both the algorithms eventually becomes equal (i.e., 96.25 percent) when s is at a sufficient level, in this case $s > 10^4$. Hence, our algorithm does not degrade the performance of the traditional classification.

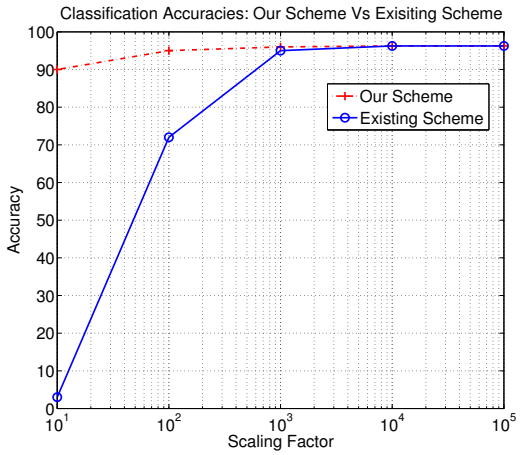

Fig. 3. Relation between scaling factor and the classification accuracy. Accuracy of both the algorithms approaches to the maximum when $s \geq 10^4$.

A. Computational complexity

In order to demonstrate the computational efficiency of the proposed method, we compare our algorithm against the algorithm in [7]. The pioneering work in [7] was the first-known PP FR work which is based on Paillier cryptography. In order to compare these works, without loss of generality, let us set the security parameter to 1024 bits and denote the computational time (in ms) for multiplication, modulo exponentiation in 1024 bits field as C_m and C_e, respectively. Table II compares both the schemes in terms of the number of modulo exponentiations and multiplications. Note that our scheme does not require any computationally-expensive modulo exponentiations. From [8], we can roughly estimate $C_e \approx 240C_m$. Let us define computational efficiencies of the proposed algorithm against [7] at the client side and the server side as e_C and e_S, respectively where

$$e_C = \frac{Complexity\ for\ client\ in\ [7]}{Complexity\ for\ client\ in\ our\ algorithm},$$
$$e_S = \frac{Complexity\ for\ server\ in\ [7]}{Complexity\ for\ server\ in\ our\ algorithm}.$$

Table III shows the computational efficiency of the proposed algorithm for different set of parameters. In Table III, we calculate the efficiency at both the client and server side by varying the parameters m, N, and n. The computational complexity to the client in the proposed algorithm is almost 120 times less than the complexity required for [7]. At the server side, our algorithm outperforms when the total number of training images (N) is less than 300. When N increases, the efficiency of the proposed algorithm at the server side drops slightly compared to [7], however efficiency at the client side is constant irrespective of any parameters. If we consider the same parameters used in numerical analysis (i.e., $N = 320$, $n = 10304$, and $m = 12$) then both schemes share the same order of computational complexity at the server side. However, the computational efficiency at the client side in the proposed algorithm is 120 times higher than [7].

B. Communication complexity

We measure the total communication complexity in terms of data being communicated between the server and client.

TABLE III
COMPARISON OF COMPUTATIONAL EFFICIENCY OF THE PROPOSED ALGORITHM AGAINST [7] .

	$n = 10000$		$n = 20000$		$n = 30000$	
(m, N)	e_C	e_S	e_C	e_S	e_C	e_S
$(10, 200)$	119	1.42	119	1.37	120	1.35
$(20, 200)$	119	1.36	119	1.31	120	1.29
$(10, 400)$	118	0.76	119	0.71	119	0.69
$(20, 400)$	118	0.73	119	0.69	119	0.66
$(10, 800)$	116	0.43	118	0.38	119	0.36
$(20, 800)$	116	0.41	118	0.37	118	0.35

In our algorithm, the client and server interact via only 8-bit data during the normalization and projection steps. Afterwards, they interact via the PP secure two-party algorithm proposed in Table I. The client initially sends 2×1024-bits to the server. Then the server sends back $5 \times 1024 \times 12 \times 320$ (i.e., $N = 320$, and $m = 12$) size of data to the client. Finally, in order to find the identity of the matched image, the client sends 1024×320-bits to the server which sends back another 1024-bits to the client. In total, the communication bandwidth required for our algorithm is $2.499MB$. However, the communications complexity of the Paillier cryptography based FR algorithm [7] requires $(n+m+1) \times 1024 = (10304+12+1) \times 1024 = 1.32MB$ bandwidth for the normalization and projection steps. For the match finding steps, [7] requires $6N + N(2l + 1) \times 1024 = 6 \times 320 + 320 \times (106 + 1) \times 1024 = 4.63MB$. In total, [7] requires 6MB of bandwidth, when security parameter is 1024, which is nearly 2.5 times higher than the proposed approach.

VI. RELATED WORKS

There were several classification algorithms developed in pattern recognition and machine learning for different applications [9]. However, only a few of them have been redesigned for PP classification in literature ([7], [10]–[14] and references therein). Majority of the work in the literature were developed for the distributed setting where different parties hold parts of the training data sets and securely train a common classifier without each party needing to disclose its own training data to other parties [13], [14]. The works in [13], [14] exploited the secure multi-party integer summation in order to compute the kernel matrix. Basically, each party generates a Gramm matrix using scalar products of training and test data samples. This Gramm matrix is later revealed to the trusted third party who will compute the kernel matrix and then classify the test sample. Revealing the Gramm matrix may leak the private data and, therefore, privacy cannot be entirely preserved.

Recently, Yuan et.al. proposed an efficient biometric identification based on random matrix operations and without homomorphic encryption and garbled circuit [16]. In fact, the work in [16] is about biometric identification (our work is about biometric recognition). In our method, the classification result and test image are known only to the client while [16] reveals the result and the test sample to another party who is using the cloud computing.

PP data classification algorithms suitable for a client-server model were studied in [7], [10]–[12]. We stress here that these

works are developed based on homomorphic cryptography. Efficient PP FR algorithm based on garbled circuit and oblivious transfer function was proposed in [18]. In [18], Sadeghi et al. proposed two algorithms: one is purely based on garbled circuit and other one is combination of Paillier homomorphic algorithm and garbled circuit. The work in [18] improved the complexity of [7] by nearly four times (our algorithm is 120- times faster than [7]) by exploiting efficient minimum value and minimum index garbled circuit. To the best of our knowledge, the proposed work in this paper is the first known lightweight PP FR algorithm which is suitable for outsourcing face recognition in the cloud.

VII. Conclusions

In this paper, we have proposed a lightweight FR to outsource the FR task to untrusted third-parties. We exploited randomization technique to preserve the privacy of the client and server side parameters. Since the proposed method excluded the usage of homomorphic cryptography, the computational and communication complexities were substantially reduced compare to the existing schemes. In order to validate the proposed methods, we have experimented our method on popular face image database. The experiment results show that the classification accuracy of the proposed method is same as the traditional approach, which proves reliability of the proposed method.

Appendix I

Proof of Lemma 1: Algorithm in Table I outputs $r_i + \mathbf{a}^T\mathbf{b}$ to the client. Let's assume that the server does not randomize the Euclidean distances $\bar{d}_i$ by random value $r_i,\ i = 1, \ldots, N$. In this case, a malicious user can infer the server's input vector $\mathbf{b}$ by selectively inputting binary vector e.g., if client wants to know b_1 then client can inputs vector $[1\ 0\ \ldots\ 0]^T$. However, since the server add freshly generated random value r_i every time, it is impossible for the client to infer server's vector $\mathbf{b}$.

Appendix II

Proof of Lemma 2: Let us assume that elements in vectors $\mathbf{a}$, $\lfloor s.\mathbf{b} \rceil$, and $s.r_i$ take maximum possible value. Let us denote elements in $\mathbf{a}$ can take maximum value of o_1 and elements in $\lfloor s.\mathbf{b} \rceil$ and $s.r_i$ can take maximum value of o_2. Hence, if number of elements in $\mathbf{a}$ and $\lfloor s.\mathbf{b} \rceil$ are equal to l, then output of the algorithm in Table I should be $\frac{o_2 + l.o_1.o_2}{s}$. In order to verify this, lets go through the Steps 8, 16, 21, 22, and 23 in Table I: Step 8 $\Rightarrow C_i = o_1.\alpha + c_i + z_i.\beta,\ i = 1, \ldots, l$. Step 16 $\Rightarrow D_i = o_2\alpha C_i = o_2.o_1.\alpha^2 + o_2.\alpha.c_i + o_2.\alpha.z_i.\beta,\ i = 1, \ldots, l$. Step 21 $\Rightarrow D = o_2.\alpha^2 + \sum_{i=1}^{l} D_i = o_2.\alpha^2 + l.o_1.o_2.\alpha^2 + o_2.\alpha.\sum_{i=1}^{l} c_i + o_2.\sum_{i=1}^{l} \alpha.z_i.\beta$. Since $K = \sum_{i=1}^{l}(z_i.\beta - c_i)$, and from Step 22 $\Rightarrow E = D + K \ mod\ \beta = o_2.\alpha^2 + l.o_1.o_2.\alpha^2 + o_2.\alpha.\sum_{i=1}^{l} c_i + o_2.\sum_{i=1}^{l} \alpha.z_i.\beta + \sum_{i=1}^{l}(z_i.\beta - c_i)\ mod\ \beta = o_2.\alpha^2 + l.o_1.o_2.\alpha^2 + (o_2.\alpha - 1).\sum_{i=1}^{l} c_i$. There is no further modulo reduction since $\beta > (l.o_1.o_2 + o_2 + 1).\alpha^2$ (Step 3), $o_2.\sum_{i=1}^{n} c_i < \alpha - o_2.l$ (Step 4), and

$$\begin{aligned} E &= o_2.\alpha^2 + l.o_1.o_2.\alpha^2 + (o_2.\alpha - 1).\sum_{i=1}^{l} c_i, \\ &< o_2.\alpha^2 + l.o_1.o_2.\alpha^2 + o_2.\alpha.\sum_{i=1}^{l} c_i \\ &< o_2.\alpha^2 + l.o_1.o_2.\alpha^2 + (\alpha^2 - \alpha.o_2.l) \\ &< o_2.\alpha^2 + l.o_1.o_2.\alpha^2 + \alpha^2 = l.o_1.o_2.\alpha^2 + (o_2 + 1).\alpha^2 < \beta. \end{aligned}$$

From Step 23, $E\ mod\ \alpha^2 = o_2.\alpha^2 + l.o_1.o_2.\alpha^2 + (o_2.\alpha - 1).\sum_{i=1}^{l} c_i\ mod\ \alpha^2 = (o_2.\alpha - 1).\sum_{i=1}^{l} c_i$. There is no further modulo reduction since $o_m.\sum_{i=1}^{n} c_i < \alpha - o_m.l$ (Step 4) and $E\ mod\ \alpha^2 = (o_2.\alpha - 1).\sum_{i=1}^{l} c_i < o_m.\alpha.\sum_{i=1}^{l} c_i < \alpha^2$. Hence, from Step 23, $\frac{E - (E\ mod\ \alpha^2)}{s.\alpha^2} = \frac{o_2.\alpha^2 + l.o_1.o_2.\alpha^2}{s.\alpha^2}$. The two-party computation algorithm always output correct result.

References

[1] Sundareswaran, S., Squicciarini, A. C., Lin, D.: Ensuring distributed accountability for data sharing in the cloud. In: IEEE Trans. Dependable and Secure Computing, vol. 9, no. 4, pp. 555–567. (Jul.-Aug. 2012)

[2] Pearson, S., Charlesworth, A.: Accountability as a way forward for privacy protection in the cloud. In: Proc. First Int'l Conf. Cloud Computing. (2009)

[3] Sirovich, L., and Kirby. M.: Low-Dimensional Procedure for the Characterization of Human Faces. In J. Opt. Soc. Am. A, vol. 2, pp. 519–524. (1987)

[4] Fukunaga, K.: *Introduction to Statistical Pattern Recognition*. Academic Press, second edition ed. (1990)

[5] Lu, R., and Lin, X., and Shen, X. (Sherman).: SPOC: A Secure and Privacy-Preserving Opportunistic Computing Framework for Mobile-Healthcare Emergency. In IEEE Trans. Parallel and Distributed Systems, vol. 24, no. 3, pp. 614–624. (Mar. 2013)

[6] The Database of Faces, (formerly The ORL Database of Faces). Available at http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase. html. AT&T Laboratories Cambridge.

[7] Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., Toft. T.: Privacy-preserving face recognition. In privacy enhancing technologies (PET09). LNCS, vol. 5672 pp 235–253. Springer. (2009)

[8] Huang, K.-H., Chung, Y.-F., Liu, C.-H., Lai, F., and Chen. T.-S.: Efficient Migration for Mobile Computing in Distributed Networks. In Computer Standards and Interfaces, vol. 31, no. 1, pp. 40-47. (2009)

[9] Duda, R. O., Hart, P. E., and Stork, D. G.: Pattern classification and scene analysis 2nd ed. (1995)

[10] Rahulamathavan, Y., Phan, R. C.-W., Chambers, J. A., Parish, D.: Facial expression recognition in the encrypted-domain based on local fisher discriminant analysis. In: IEEE Trans. Affective Computing, vol. 4, no. 1, pp. 83–92. (Jan.-Mar., 2013)

[11] Rahulamathavan, Y., and Veluru, S., and Phan, R.C.-W., and Chambers, J. A., and Rajarajan, M.: Privacy-Preserving Clinical Decision Support System Using Gaussian Kernel-Based Classification. In IEEE Journal of Biomedical and Health Informatics, vol. 18, no. 1, pp. 56–66. (Jan. 2014)

[12] Rahulamathavan, Y., and Phan, R.C.-W., and Veluru, S., and Cumanan, K., and Rajarajan, M.: Privacy-Preserving Multi-Class Support Vector Machine for Outsourcing the Data Classification in Cloud. In IEEE Trans. Dependable and Secure Computing, vol. 11, no. 5, pp. 467–479. (2014)

[13] Yu, H., Jiang, X., Vaidya. J.: Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data. In: Proc. ACM Symp. Applied Computing (SAC). (2006)

[14] Chen, K., Liu, L.: Privacy preserving data classification with rotation perturbation. In: ICDM, IEEE Computer Society, pp. 589–592. (2005)

[15] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238, Springer, Heidelberg. (1999)

[16] Yuan, J., Yu, S.: Efficient privacy-preserving biometric identification in cloud computing. In IEEE INFOCOM, pp. 2652-2660. (2013)

[17] Meyer, C. D.: Matrix Analysis and Applied Linear Algebra. In Society for Industrial and Applied Mathematics, isbn – 0-89871-454-0, Philadelphia, PA. (2000)

[18] Sadeghi, A.R., Schneider, T., and Wehrenberg, I.: Efficient privacypreserving face recognition. In Information, Security and Cryptology ICISC 2009. Springer Berlin Heidelberg, pp. 229-244. (2010)