Privacy Preserving Face Recognition Utilizing Differential Privacy

Mahawaga Arachchige, Pathum; Bertok, Peter; Khalil, Ibrahim; Liu, D.; Camtepe, Seyit

Privacy Preserving Face Recognition Utilizing Differential Privacy

M.A.P. Chamikara[a,b,*], P. Bertok[a], I. Khalil[a], D. Liu[b], S. Camtepe[b]

[a]*RMIT University, Australia*
[b]*CSIRO Data61, Australia*

Abstract

The published article can be found at `https://doi.org/10.1016/j.cose.2020.101951`

Facial recognition technologies are implemented in many areas, including but not limited to, citizen surveillance, crime control, activity monitoring, and facial expression evaluation. However, processing biometric information is a resource-intensive task that often involves third-party servers, which can be accessed by adversaries with malicious intent. Biometric information delivered to untrusted third-party servers in an uncontrolled manner can be considered a significant privacy leak (i.e. uncontrolled information release) as biometrics can be correlated with sensitive data such as healthcare or financial records. In this paper, we propose a privacy-preserving technique for "controlled information release", where we disguise an original face image and prevent leakage of the biometric features while identifying a person. We introduce a new privacy-preserving face recognition protocol named PEEP (Privacy using EigEnface Perturbation) that utilizes local differential privacy. PEEP applies perturbation to Eigenfaces utilizing differential privacy and stores only the perturbed data in the third-party servers to run a standard Eigenface recognition algorithm. As a result, the trained model will not be vulnerable to privacy attacks such as membership inference and model memorization attacks. Our experiments show that PEEP exhibits a classification accuracy of around 70% - 90% under standard privacy settings.

Keywords: Privacy preserving face recognition, differential privacy, face recognition, privacy in artificial intelligence, privacy preserving machine learning

1. Introduction

Face recognition has many applications in the fields of image processing and computer vision; advancements in related technologies allow its efficient and accurate integration in many areas from

*Corresponding author
Email address: `pathumchamikara.mahawagaarachchige@rmit.edu.au` (M.A.P. Chamikara)

individual face recognition for unlocking a mobile device to crowd surveillance. Companies have also invested heavily in this field; Google's facial recognition in the Google Glass project [1], Facebook's DeepFace technology [2], and Apple's patented face identification system [3] are examples of the growing number of facial identification systems. Existing face recognition technologies and the widespread use of biometrics introduce a serious threat to individuals' privacy, exacerbated by the fact that biometric identification is often done quietly, without proper consent from observed people. For example, the UK uses an estimated 4.2 million surveillance cameras to monitor public areas [4]. However, it is not feasible to obtain explicit consent from an extremely large number of persons being watched. Nevertheless, facial images directly reflect the owners' identity, and they can be easily linked to other sensitive information such as health records and financial records, raising privacy concerns. Biometric data analysis systems often need to employ high-performance third-party servers to conduct complex computational operations on large numbers of biometric data inputs. However, these third-party servers can be accessed by untrusted parties causing privacy issues.

Among different definitions, information privacy can be defined as the "controlled information release" that permits an anticipated level of utility via a private function that protects the identity of the data owners [5]. Privacy-preserving face recognition involves at least two main parties: one needs to recognize an image (party 1), and the other holds the database of images (party 2). Data encryption would allow party 1 to learn the result without learning the execution of the recognition algorithm or its parameters, whereas party 2 would not learn the input image or the result of the recognition process [4]. However, the high computational complexity and the need to trust the parties for their respective responsibilities can be major issues. Proposed in this paper is data perturbation, which is significantly less computationally complex, but incurs a certain level of utility loss. Data perturbation allows all parties to be untrusted [6]. The parties will learn only the classification result (e.g. name/tag of the image) with a certain level of confidence, but will not have access to the original image. The literature identifies two major application scenarios of recognition technologies in which a third party server is used. They are (1) the use of biometric data such as face images and fingerprint to identify and authenticate a person (e.g. at border crossings) and (2) deploy surveillance cameras in public places to automatically match or identify faces (offender tracking/criminal investigations [7]). There are a few methods that are based on encryption to provide privacy-preserving face recognition [4, 8, 9], which need one or more trusted third parties in a server-based setting (e.g. cloud servers). However, in an environment where no trusted party is present, such semi-honest approaches raise privacy concerns, as the authorized trusted parties are still allowed to access the original image data (raw or encrypted). Moreover, an encryption-based mechanism for scenarios that process millions of faces would be extremely inefficient and difficult to maintain. The methods such as $k - same$ [10] for preserving privacy by de-identifying face images can avoid the necessity of a trusted third-party. However, such methods introduce utility issues in large scale scenarios with millions of faces, due to the limitations of the underlying privacy models used (e.g. $k - anonymity$) [6]. We identify five main types of issues (TYIS) with the existing privacy-preserving approaches for face recognition. They are as follows. TYIS 1: face biometrics should not be linkable to other sensitive data, TYIS 2: the method should be scalable and resource friendly, TYIS 3: face biometrics should not be accessible by anyone (i.e. use one-way transformation), TYIS 4: face biometrics of the same person from two different applications should not be linkable, and TYIS 5: face biometrics should be revocable (if data is leaked, the application should have a way of revoking them to prevent any malicious use).

This paper proposes a method to control privacy leakage from face recognition, answering the five TYIS better than the existing privacy-preserving face recognition approaches. We propose an approach that stores data in a perturbed form. The method utilizes differential privacy to devise a novel technique (named PEEP: Privacy using EigEnface Perturbation) for privacy-preserving face recognition. PEEP uses the properties of local differential privacy to apply perturbation on input image data to limit potential privacy leaks due to the involvement of untrusted third-party servers and users. To avoid the necessity of a trusted third party, we apply randomization to the data used for training and testing. Due to the extremely low complexity, PEEP can be easily implemented on resource-constrained devices, allowing the possibility of perturbation at the input end. The ability to control the level of privacy via adjusting the privacy budget is an additional advantage of the proposed method. The privacy budget is used to signify the level of privacy provided by a privacy-preserving algorithm; the higher the privacy budget, the lower the privacy. PEEP utilizes local differential privacy at the cost of as low as 6 percent drop in accuracy (e.g. 85% to 79%) with a privacy budget of $\varepsilon = 8$. A mechanism with a privacy budget (ε) of $0 < \varepsilon \leq 9$ is considered to provide an acceptable level of privacy [11, 12]. Consequently, PEEP is capable of adjusting the privacy-accuracy trade-off by changing the privacy budget through added noise.

The rest of the paper is organized as follows. Section 2 provides a summary of existing related work. The foundations of the proposed work are briefly discussed in Section 3. Section 4 provides the technical details of the proposed approach. The results are discussed in Section 5. The paper is concluded in Section 6.

2. Related Work

Literature shows a vast advancement in the area of face recognition that has employed different approaches, such as input image preprocessing [13], statistical approaches [14, 15], and deep learning [16]. The continuous improvements in the field have significantly improved the accuracy of face recognition making it a vastly used approach in many fields [16]. Furthermore, the approaches, such as proposed by Cendrillon et al., show the dynamic capabilities of face recognition approaches that allow real-time processing [17]. However, biometric data analysis is a vast area not limited to face recognition. With biometric data, a major threat is privacy violation [18]. Biometric data are almost always non-revocable and can be used to identify a person in a large set of individuals easily; hence, it is essential to apply some privacy-preserving mechanism when using biometrics, e.g. for identification and authentication [19]. Literature shows a few approaches to address privacy issues in face recognition. Zekeriya Erkin et al. (ZEYN) [4] introduced a privacy-preserving face recognition method based on a cryptographic protocol for comparing two Pailler-encrypted values. Their solution focuses on a two-party scenario where one party holds the privacy-preserving algorithm and the database of face images, and the other party wants to recognize/classify a facial image input. ZEYN requires O(log M) rounds, and it needs computationally expensive operations on homomorphically encrypted data to recognize a face in a database of images, hence not suitable for large scale scenarios. Ahman-Reza Sadehi et al. (ANRA) [8] introduced a relatively efficient method based on homomorphic encryption with garbled circuits. Nevertheless, the complexity of ANRA also has the same problem of failing to address large scale scenarios. Xiang et al. tried to overcome the computational complexities of the previous methods by introducing another cryptographic mechanism that uses the cloud [9] for outsourced computations. However, being a semi-honest model, introducing another untrusted module such as the cloud increases the possibility of privacy leak. PE-MIU (Privacy-Enhancing face recognition approach based on Minimum Information Units) [20] and POR (lightweight privacy-preserving adaptive boosting (AdaBoost) classification framework for face recognition) [21] are two other recently developed privacy-preserving face recognition approaches. PE-MIU is based on the concept of minimum information units, whereas POR is based on additive secret sharing. PE-MIU is also a semi-honest approach, which lacks a proper privacy definition in its mechanism. Moreover, the scalability of PE-MIU can be limited due to the exponential template comparisons necessary during the execution of the proposed algorithm. POR provides a relatively efficient approach compared to the previous encryption-based approaches. However, being a semi-honest approach, POR inherits the issues of any semi-honest approach discussed above. The proposed cryptographic methods cannot work without a trusted third party, and these trusted parties may later behave maliciously. Newton et al. proposed a de-identification approach for face images (named as $k - same$), which does not need complex cryptographic operations [10]. The proposed method is based on $k-anonymity$ [6, 22]. However, $k-anonymity$ tends to reduce accuracy and increase information leak when introduced with high dimensional data [6]. The same problem can occur when using $k - same$ for large scale scenarios involving the surveillance of millions of people. In addition to these works, researchers have looked at complementary techniques such as developing privacy-friendly surveillance cameras [23, 24], but these methods do not provide sufficient accuracy for privacy-preserving face recognition.

Fingerprint data and iris data are two other heavily used biometrics for identification and authentication. Privacy-preserving finger code authentication [25], and privacy-preserving key generation for iris biometrics [26] are two approaches that apply cryptographic methods to maintain the privacy of fingerprint and iris data. However, these solutions also need more efficient procedures, as cryptographic approaches are inefficient in calculations [26, 27]. Privacy-preserving fingerprint and iris analysis can be possible future applications for PEEP, but this needs further investigation. Classification is the most commonly applied data mining technique that is used in biometric systems [28]. Encryption and data perturbation are two main approaches also used for privacy-preserving data mining (PPDM) [29]. Data perturbation often entails lower computational complexity than encryption at the expense of utility. Hence, data perturbation is better at producing high efficiency in large scale data mining. Noise addition, geometric transformation, randomization, condensation, and hybrid perturbation are a few of the perturbation approaches [30, 6]. As data perturbation methods do not change the original input data formats, they may concede some privacy leak [31]. A privacy model defines the constraints on the level of privacy of a particular perturbation mechanism [31]; $k - anonymity$, $l - diversity$, $(\alpha, k) - anonymity$, $t - closeness$ and differential privacy (DP) are some of such privacy models [6]. DP was developed to provide a better level of privacy guarantee compared to previous privacy models that are vulnerable to different privacy attacks [32, 33]. Laplace mechanism, Gaussian mechanism [34], geometric mechanism, randomized response [35], and staircase mechanism [36] are a few of the fundamental mechanisms used to achieve DP. There are many practical examples where these fundamental mechanisms have been used to build differentially private algorithms/methods. LDPMiner

[35], PINQ [37], RAPPOR [38], and Deep Learning with DP [11] are a few examples of such practical applications of DP.

3. Foundations of Differential Privacy and Eigenface recognition

In this section, we describe the background of the techniques used in the proposed solution. PEEP conducts privacy-preserving face recognition utilizing the concepts of differential privacy and eigenface recognition.

3.1. Differential Privacy (DP)

DP is a privacy model that is known to render maximum privacy by minimizing the chance of individual record identification [36]. In principle, DP defines the bounds to how much information can be revealed to a third party/adversary about someone's data being present in a particular database. Conventionally ε (epsilon) is used to denote the level of privacy rendered by a randomized privacy-preserving algorithm ($\mathcal{M}$) over a particular database ($\mathcal{D}$); ε is called the privacy budget that provides an insight into the privacy loss of a DP algorithm. The higher the value of ε, the higher the privacy loss.

Let us take two adjacent datasets of $\mathcal{D}$, x and y, where y differs from x only by (plus or minus) one person. Then $\mathcal{M}$ satisfies (ε)-DP if Equation (1) holds. Assume, datasets x and y as being collections of records from a universe $\mathcal{X}$ and $\mathbb{N}$ denotes the set of all non-negative integers including zero.

Definition 1. *A randomized algorithm $\mathcal{M}$ with domain $\mathcal{N}^{|\mathcal{X}|}$ and range R: is ε-differentially private if for every adjacent x, $y \in \mathcal{N}^{|\mathcal{X}|}$ and for any subset $\mathcal{S} \subseteq \mathcal{R}$*

$$\mathcal{P}r[(\mathcal{M}(x) \in \mathcal{S})] \leq \exp(\varepsilon)\ \mathcal{P}r[(\mathcal{M}(y) \in \mathcal{S})] \tag{1}$$

3.2. Global vs. Local Differential Privacy

Global differential privacy (GDP) and local differential privacy (LDP) are the two main approaches to DP. In the GDP setting, there is a trusted curator who applies carefully calibrated random noise to the real values returned for a particular query. The GDP setting is also called the trusted curator model [39]. Laplace mechanism and Gaussian mechanism [40] are two of the most frequently used noise generation methods in GDP [40]. A randomized algorithm, $\mathcal{M}$ provides ε-GDP if Equation (1) holds. LDP randomizes data before the curator can access them, without the need of a trusted curator. LDP is also called the untrusted curator model [36]. LDP can also be used by a trusted party to randomize all records in a database at once. LDP algorithms may often produce too noisy data, as noise is applied to achieve individual record privacy. LDP is considered to be a strong and rigorous notion of privacy that provides plausible deniability and deemed to be a state-of-the-art approach for privacy-preserving data collection and distribution. A randomized algorithm $\mathcal{A}$ provides ε-LDP if Equation (2) holds [38].

Definition 2. *A randomized algorithm $\mathcal{A}$ satisfies ε-LDP if for all pairs of users' inputs v_1 and v_2 and for all $\mathcal{Q} \subseteq Range(\mathcal{A})$, and for ($\varepsilon \geq 0$) Equation* (2) *holds. $Range(\mathcal{A})$ is the set of all possible outputs of the randomized algorithm $\mathcal{A}$.*

$$\mathcal{P}r[\mathcal{A}(v_1) \in \mathcal{Q}] \leq \exp(\varepsilon)\ Pr[\mathcal{A}(v_2) \in \mathcal{Q}] \tag{2}$$

3.3. Sensitivity

Sensitivity is defined as the maximum influence that a single individual can have on the result of a numeric query. Consider a function f, the sensitivity (Δf) of f can be given as in Equation (3) where x and y are two neighboring databases (or in LDP, adjacent records) and $\|.\|_1$ represents the $L1$ norm of a vector [41].

$$\Delta f = max\{\|f(x) - f(y)\|_1\} \tag{3}$$

3.4. Laplace Mechanism

The Laplace mechanism is considered to be one of the most generic approaches to achieve DP [40]. Laplace noise can be added to a function output ($\mathcal{F}(\mathcal{D})$) as given in Equation 5 to produce a differentially private output. Δf denotes the sensitivity of the function f. In local differentially private setting, the scale of the Laplacian noise is equal to $\Delta f/\varepsilon$, and the position is the current input value ($\mathcal{F}(\mathcal{D})$).

$$\mathcal{PF}(\mathcal{D}) = \mathcal{F}(\mathcal{D}) + Lap(\frac{\Delta f}{\varepsilon}) \tag{4}$$

$$\mathcal{PF}(\mathcal{D}) = \frac{\varepsilon}{2\Delta f}\ e^{-\frac{|x-\mathcal{F}(\mathcal{D})|\varepsilon}{\Delta f}} \tag{5}$$

3.5. Eigenfaces and Eigenface recognition

The process of face recognition involves data classification where input data are images, and output classes are persons' names. A face recognition algorithm needs to be first trained with an existing database of faces. The trained model will then be used to recognize a person's name using an image input. The training algorithm often needs various images to have high accuracy. When the model needs to be trained to recognize a large number of persons, the training algorithm also needs a large number of training images. Image data are often large, and the higher the number of faces to be trained, the slower the algorithm. However, facial recognition systems need high efficiency, as many of them are employed in real-time systems such as citizen surveillance [42]. When an artificial neural network (ANN) is used for face recognition, the input images need to be flattened into 1-d vectors. An image with the dimensions $m \times n$ will result in an $mn \times 1$ vector. High-resolution images will result in extremely long 1-d vectors, which leads to slow training and testing of the corresponding ANN. Dimensionality reduction methods can be used to avoid such complexities, and allow face recognition to concentrate on the essential features, and to ignore the noise in the input images. In dimensionality reduction, the points are projected onto a higher-dimensional line, which is named as a hyperplane. Principal component analysis (PCA) is a dimensionality reduction technique that represents a hyperplane with maximum variance. This hyperplane can be determined using eigenvectors, which can be computed using the covariance matrix of input data [42].

Algorithm 1: Generating Eigenfaces

```
Input:  $\{x_1^c, \dots, x_n^c\}$ ← normalized and centered examples
        $nc$                      ← expected number of PCA components
Output: $\mathcal{EIMAT}$          ← matrix of eigenfaces
1 for each $x_i^c$ do
2     flatten $x_i^c$ to produce vector $t_i$
3 compute the mean face vector ($\mathcal{F}_m$), $\mathcal{F}_m = \frac{1}{n}\Sigma_{i=1}^{n} t_i$;
4 for each $x_i^c$ do
5     $s_i = t_i - \mathcal{F}_m$;
6 generate covariance matrix, $\mathcal{C}$,
  $\mathcal{C} = \frac{1}{n}\Sigma_{i=1}^{n} s_i \times s_i^{\mathcal{T}} = \mathcal{A}\mathcal{A}^{\mathcal{T}}$, where, $\mathcal{A} = [s_1 s_2 \dots s_n]$;
7 calculate the eigenvectors $e_i$ of $\mathcal{A}\mathcal{A}^T$
  since, $\mathcal{A}\mathcal{A}^T$ can be extensive, derive $e_i$ from the eigenvectors $u_i$ of $\mathcal{A}^{\mathcal{T}}\mathcal{A}$, where, $e_i = \mathcal{A}u_i$;
8 compute the $n$ best eigenvectors $e_i$ such that, $\|e_i\| = 1$;
9 return $nc$ eigenvectors which corresponds to the $nc$ largest eigenvalues
```

Algorithm 1 shows the steps for generating Eigenfaces. As shown in the algorithm, an eigenface [43] utilizes PCA to represent a dimensionality-reduced version of an input image. A particular eigenface considers a predefined number of the largest eigenvectors as the principal axes that we project our data on to, hence producing reduced dimensions [42]. We can reduce the dimensions of an $m \times n$ image into a k dimensional eigenface where k is the largest k eigenvectors. By doing this, we can consider only the most essential characteristics of an input image and increase the speed of a facial recognition algorithm while preserving high accuracy. Equation 6 provides the mathematical representation of an eigenface where $\mathcal{F}$ is a new face, $\mathcal{F}_m$ is the mean or the average face, $\mathcal{F}_i$ is an EigenFace, and α_i are scalar multipliers which we have to choose in order to create new faces.

$$\mathcal{F} = \mathcal{F}_m + \sum_{i=1}^{n} \alpha_i \mathcal{F}_i \tag{6}$$

4. Our Approach: PEEP

In this section, we discuss the steps employed in the proposed privacy-preserving face recognition approach (named as PEEP). We utilize DP to apply confidentiality to face recognition. PEEP applies randomization upon the eigenfaces to create privacy-preserving versions of input images. We assume that any input device used to capture the facial images uses PEEP to apply randomization before sending the images to the storage devices/servers.

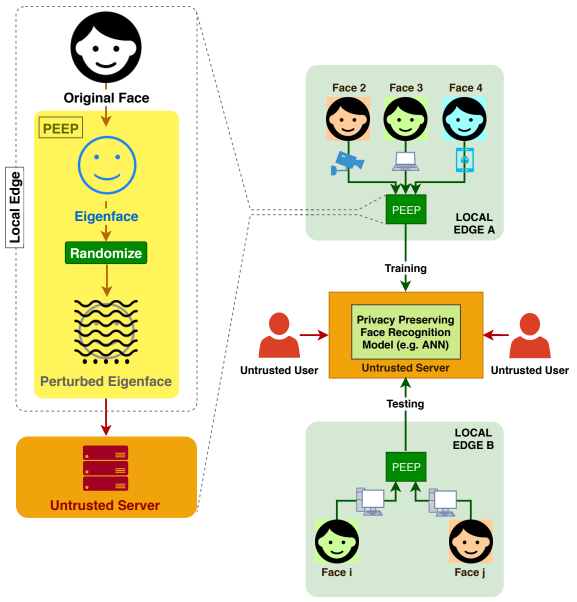

Figure 1: Privacy-preserving face recognition using PEEP. The figure shows the placement of PEEP in a face recognition system. As shown, PEEP randomizes both training and testing images so that the untrusted third-party servers do not leak any private data to untrusted users. The callout figure in the left-hand side shows the basic flow of randomization inside PEEP, which applies Laplacian noise over eigenfaces.

As depicted by the callout box in Figure 1, PEEP involves three primary steps to enforce privacy on face recognition. They are, 1. accepting original face images, 2. generating eigenfaces, and 3. adding Laplacian noise to randomize the images. In the proposed setting, the face recognition model (e.g. MLPClassifier) will be trained solely using randomized data. In this setup, an untrusted server will hold only a privacy-preserving version of the face recognition model.

4.1. Distributed eigenface generation

When the number of input faces increases to a large number, it is important that the eigenface calculation (generation) can be distributed in order to maintain efficiency. Algorithm 2 shows an incremental calculation approach of eigenfaces where a central computer (CC) in the local edge contributes to the calculation of eigenfaces in a distributed fashion. As shown in step 5 in Algorithm 2, the mean face vectors, $\mathcal{F}_m^i$ that are generated for each partition of input data are collected and merged (using Equation 7) by the CC to generate the global mean face vector $\mathcal{F}_m^{glob}$. Similarly, the CC generates the global covariance matrix, $\mathcal{C}^{glob}$ (refer step 10 Algorithm 2) using the covariance matrices generated for each partition using Equation 10. In this way, PEEP manages to maintain the efficiency of eigenface generation for extensive datasets.

$$\mathcal{F}_m^{glob} = \begin{bmatrix} \frac{m_1 \times \overline{y_{11}} + m_2 \times \overline{y_{12}} + \ldots + m_k \times \overline{y_{1k}}}{m_1 + m_2 + \ldots + m_k} \\ \frac{m_1 \times \overline{y_{21}} + m_2 \times \overline{y_{22}} + \ldots + m_k \times \overline{y_{2k}}}{m_1 + m_2 + \ldots + m_k} \\ \vdots \\ \frac{m_1 \times \overline{y_{n1}} + m_2 \times \overline{y_{n2}} + \ldots + m_k \times \overline{y_{nk}}}{m_1 + m_2 + \ldots + m_k} \end{bmatrix}_{n \times 1} \tag{7}$$

In Equation 7, m_i refers to the number of eigenfaces in the i^{th} partition, whereas $\overline{y_{ij}}$ refers to the mean of the j^{th} index of the i^{th} partition. To merge the covariance matrices, the pairwise covariance update formula introduced in [44] is adapted as shown in Equation 10 [45]. The pairwise covariance update formula for the two merged two column (u and v) data partitions, A and B, can be written as shown in Equation 8 where the merged dataset is denoted as X.

$$Cov(X) = \frac{\frac{C_A}{(m_A - 1)} + \frac{C_B}{(m_B - 1)} + (\mu_{u,A} - \mu_{u,B})(\mu_{v,A} - \mu_{v,B}).\frac{m_A.m_B}{m_X}}{(m_X - 1)} \tag{8}$$

Where, $\mu_{u,A}, \mu_{u,A}, \mu_{v,A}, \mu_{v,B}$ are means of u and v of the two data partitions A and B, respectively. C_A and C_B are the co-moments of the two data partitions A and B where the co-moment of a two column (u and v) dataset D is represented as,

$$C_D = \sum_{(u,v) \in D} (u - \mu_u)(v - \mu_v) \tag{9}$$

Therefore, the variance-covariance matrix update formula of the two data partitions D_g and D_i can be written as shown in Equation 10,

$$\mathcal{C}^{glob} = \frac{\frac{\mathcal{C}^{glob}}{(m_{D_g} - 1)} + \frac{\mathcal{C}_i}{(m_{D_i} - 1)} + (\mu_{D_g}(MI_g) - \mu_{D_i}(MI_g))(\mu_{D_g}(MI_i) - \mu_{D_i}(MI_i)).\frac{m_{D_g}.m_{D_i}}{m_{D_{new}}}}{(m_{D_{new}} - 1)} \tag{10}$$

In Equation 10, assume that $\mathcal{C}^{glob}$ and $\mathcal{C}_i$ are the covariance matrices returned for the data partitions D_g and D_i respectively, where D_g represents the global partition (concatenation of all the former partition), whereas D_i represents the new partition introduced to the calculation. D_{new} is the merged dataset of the the data partitions, D_g and D_i. μ_{D_g} and μ_{D_i} are mean vectors of D_g and D_i respectively. m_D represents the number of eigenfaces in the corresponding dataset. Equation 10 will be iteratively calculated for all the data partitions to generate the final value of D_g. $\mathcal{C}^{glob}$ is initialized with the first partition, and D_i will start from the second partition and,

$$MI_i = \begin{bmatrix} [1]_n \\ [2]_n \\ [3]_n \\ \vdots \\ [n]_n \end{bmatrix}_{n \times n} \tag{11}$$

We can also run Algorithm 2 in distributed computing nodes (DCN) within the local edge to conduct efficient eigenface generation. In such a setting, DCNs will communicate with a central computer (in the local edge) to generate the global mean face ($\mathcal{F}_m^{glob}$) and the global covariance matrix ($\mathcal{C}^{glob}$). In this way, an agency can deal with a large number of input faces by maintaining a feasible number of DCNs.

Algorithm 2: Incremental calculation of Eigenfaces using data partitions

```
Input:  $\{x_1^{pk}, \ldots, x_n^{pk}\}$ ← normalized and centered example partition, $pk$
        $nc$ ← expected number of PCA components
Output: $\mathcal{EIMAT}$ ← matrix of eigenfaces
1  for each $x_i^{pk}$ do
2      flatten $x_i^{pk}$ to produce vector $t_i$
3  compute the mean face vector ($\mathcal{F}_m^i$), $\mathcal{F}_m^i = \frac{1}{n}\Sigma_{i=1}^n t_i$;
4  collect $\mathcal{F}_m^i$ at a central computer (CC) in the local edge ;
5  receive global mean face vector, $\mathcal{F}_m^{glob}$ from the CC;
6  for each $x_i^c$ do
7      $s_i = t_i - \mathcal{F}_m^{glob}$;
8  generate covariance matrix, $\mathcal{C}_i$,
     $\mathcal{C}_i = \frac{1}{n}\Sigma_{i=1}^n s_i \times s_i^{\mathcal{T}} = \mathcal{A}_i\mathcal{A}_i^{\mathcal{T}}$, where, $\mathcal{A}_i = [s_1 s_2 \ldots s_n]$;
9  collect $\mathcal{C}_i$ at the CC;
10 receive global covariance matrix, $\mathcal{C}^{glob}$ from the CC;
11 calculate the eigenvectors $e_i$ of $\mathcal{A}\mathcal{A}^T$, where $\mathcal{C}^{glob} = \mathcal{A}\mathcal{A}^T$
     since, $\mathcal{A}\mathcal{A}^T$ can be extensive, derive $e_i$ from the eigenvectors $u_i$ of $\mathcal{A}^{\mathcal{T}}\mathcal{A}$, where, $e_i = \mathcal{A}u_i$;
12 compute the $n$ best eigenvectors $e_i$ such that, $\|e_i\| = 1$;
13 return $nc$ eigenvectors which corresponds to the $nc$ largest eigenvalues
```

4.2. Generation of the principal components

After accepting the image inputs, PEEP normalizes the images to match a predefined resolution (which is accepted by PEEP as an input). We consider a default resolution normalization of 47×62. However, based on the input image sizes and the computational power of the edge devices, the users can increase or decrease the values of irw and irh suitably. Following the steps of Algorithm 1, PEEP calculates the principal components by considering the eigenvectors using the corresponding covariance matrix. The largest nc (the number of principal components) number of eigenvectors are used to create a particular eigenface (nc is taken as input). The higher the nc, the higher the representation of input features, the lower the efficiency. It is important to select a suitable number for nc that can provide high accuracy and high efficiency simultaneously. A reliable number for nc can be determined by investigating the change in the trained model's accuracy.

4.3. Declaring the sensitivity before noise addition

PEEP scales the indices of the identified PCA vectors within the interval [0,1] as the next step after generating the eigenfaces. In LDP, the sensitivity is the maximum difference between two adjacent records. In PEEP, the inputs are images, and each image is dimensionality reduced to form a vector by using PCA (PCA_vectors). As PEEP adds noise to these vectors (PCA_vectors), the sensitivity of PEEP is the maximum difference between two such PCA_vectors which can be denoted by Equation 12, where $\mathcal{FSV}^j$ represents a flattened image vector scaled within the interval [0,1], $\mathcal{FSV}^{j+1}$ is adjacent to $\mathcal{FSV}^j$. Since PEEP examines the Cartesian system, we can consider the maximum Euclidean distance for the sensitivity, which is equal to a maximum of $\sqrt{nc}$ where nc is the number of principal components. As the normalized PCA_vectors are bounded by 0 and 1, a sensitivity much greater than 1 would entail a substantial level of noise, which can reduce the utility drastically as we use LDP for the noise application mechanism. Hence, we select the sensitivity to be the maximum difference between two indices, which is equal to 1. Now the scale of the Laplacian noise will be equal to $1/\varepsilon$. As future work, we are conducting further algebraic analysis of sensitivity to improve the precision and flexibility of the Laplace mechanism in the proposed approach of face recognition. After defining the position and scale parameters, PEEP adds Laplacian noise to each index of PCA_vectors. We take the position of the noise to be the index values and the scale of the noise to be $1/\varepsilon$. To generate the private versions of images ($\mathcal{PI}$), we can perturb each index according to Equation 13, where $\mathcal{FSV}_i$

represents an index of the flattened image vectors scaled within the interval [0,1].

$$\Delta f = max\{\|\mathcal{FSV}^{j} - \mathcal{FSV}^{(j+1)}\|_1\} \tag{12}$$

4.4. Introducing Laplacian noise

After defining the position and scale parameters, PEEP adds Laplacian noise to each index of PCA_vectors. We take the position of the noise to be the index values and the scale of the noise to be $1/\varepsilon$. To generate the private versions of images ($\mathcal{PI}$), we perturb each index according to Equation 13, where FSV_i represents an index of the flattened image vectors scaled between 0 and 1. The user can provide a suitable ε value depending on the amount of privacy required and after considering the following guidelines. The higher the ε value, the lower the privacy. As a norm, $0 < \varepsilon \leq 9$ is considered as an acceptable level of privacy [11]. We follow the same standard and use an upper limit of 9 for ε.

$$\mathcal{PI} = \frac{\varepsilon}{2\Delta f}\, e^{-\frac{|x-\mathcal{FSV}_i|\varepsilon}{\Delta f}} \tag{13}$$

Algorithm 3: Differentially private facial recognition: PEEP

```
Input:
    $\{x_1, \ldots, x_n\}$  $\leftarrow$  examples
    $imthresh$  $\leftarrow$  number of images per face
    $\varepsilon$  $\leftarrow$  privacy budget
    $irw$  $\leftarrow$  pixel width (default = 47)
    $irh$  $\leftarrow$  pixel height (default = 62)
    $nc$  $\leftarrow$  number of PCA components
Output:
    $\mathcal{DPFRS}$  $\leftarrow$  privacy preserving
                facial recognition model
1  Find the minimum width of all image ($w_{min}$);
2  Fine the minimum height of all image ($h_{min}$);
3  if $irw < w_{min} \lor irh < h_{min}$ then
4      $irw = w_{min}$
5      $irh = h_{min}$
6  normalize the example resolution to $irw \times irh$ ;
7  if $nc > irw \lor nc > irh$ then
8      $nc = min(irw, irh)$
9  generate the flattened vectors ($v_i$) for each $x_i$;
10 generate the first $nc$ PCA components ($\mathcal{PCA}_i$) for each input, $v_i$, according to Algorithm 1;
11 scale all the indices of $v_i$ between 0 and 1 to generate $sv_i$;
12 apply $\frac{\varepsilon}{2\Delta f}\, e^{-\frac{|x-\mathcal{FSV}_i|\varepsilon}{\Delta F}}$ to each index of $sv_i$ with $sensitivity$ $(\Delta f) = 1$ ;
13 feed $\{sv_1, \ldots, sv_n\}$ and corresponding targets to the classification model;
14 train the classification model using the randomized data to produce a differentially private classification
     model ($\mathcal{DPFRS}$);
15 release the $\mathcal{DPFRS}$;
```

4.5. Algorithm for generating a differentially private face recognition model

Algorithm 3 shows the steps of PEEP in conducting privacy-preserving face recognition model training. As shown in the algorithm, *irw* and *irh* parameters are used to increase the resolution of the input images. We use the input parameter, *imthresh*, to accept the number of images considered per single face (person). Since the main task of face recognition is image classification, each face represents a class. In order to produce good accuracy, a classification model should have a good image representation. Consequently, *imthresh* is a valuable parameter that directly influences the accuracy, where a higher value of *imthresh* will certainly contribute to higher accuracy due to the better representation of images between the classes (faces). Hence, *imthresh* allows the algorithm to extract eigenfaces that provide a better representation of the input images resulting in better accuracy. Step 3 makes sure that the number of PCA components selected does not go beyond the allowed threshold.

4.6. Privacy preserving face recognition using PEEP

As shown in Figure 1, each image input will be subjected to PEEP randomization before training or testing. The Eigenface generation and randomization take place within the local edge bounds. We assume that all input devices communicate with the third party servers only through PEEP, and the face recognition database stores only the perturbed images. Since the face recognition model (e.g. MLPClassifier) is trained only using perturbed images (perturbed eigenfaces), the trained model will not leak private information. Any untrusted access to the server will not allow any loss of valuable biometric data to malicious third parties. Since PEEP perturbs testing data, there is minimal privacy leak from testing data (testing image inputs) as well.

4.7. Theoretical privacy guarantee of PEEP on trained classifier

Although additional computations are carried out on the outcome of a differentially private algorithm, they do not weaken the privacy guarantee. The results of additional computations on ε-DP outcome will still be ε-DP. This property of DP is called the postprocessing invariance/robustness [46]. Since PEEP utilizes DP, PEEP also inherits postprocessing invariance. The postprocessing invariance property guarantees that the trained model of perturbed data satisfies the same privacy imposed by PEEP. Therefore, the proposed method ensures that there is a minimal level of privacy leak from the third party untrusted servers. However, we further investigate the privacy strength of PEEP using empirical evidence under Section 5.

4.8. Datasets

We used the open face image dataset and the large-scale CelebFaces Attributes (CelebA) dataset (see Figure 2 for sample images) to test the performance of PEEP. Open face image dataset named lfw-funneled is available at the University of Massachusetts website named "Labeled Faces in the Wild"[1]. The lfw-funneled dataset has 13,233 gray images. We limit the minimum number of faces per person to 100, which limits the number of images to 1,140 with five classes; "Colin Powell", "Donald Rumsfeld", "George W Bush", "Gerhard Schroeder", and "Tony Blair"[2]. Figure 2 shows the appearance of 8

[1] http://vis-www.cs.umass.edu/lfw/

[2] The diversity of the classes of the dataset are as follows, "Colin Powell": 236, "Donald Rumsfeld": 121, "George W Bush": 530, "Gerhard Schroeder": 109, and "Tony Blair": 144.

sample images that are available in the datasets used. We used 70% of the input dataset for training and 30% for testing. CelebA[3] dataset has more than 200K celebrity images, each with 40 attribute annotations. CelebA has 10,177 number of identities, 202,599 number of face images, 5 landmark locations, and 40 binary attributes annotations per image.

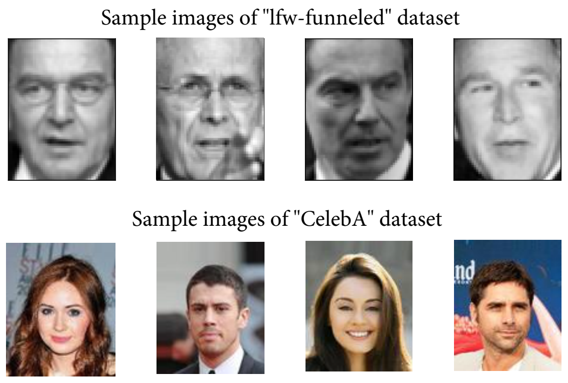
Sample images of "lfw-funneled" dataset
Sample images of "CelebA" dataset

Figure 2: Sample images of the two databases. The lfw-funneled dataset is composed of gray images whereas the CelebA dataset is composed of colored images.

4.9. Eigenfaces and Eigenface perturbation

Figure 3 shows 8 sample eigenfaces before perturbation. As the figure shows, eigenfaces already hide some features of the original images due to the dimensionality reduction [47]. However, eigenfaces alone would not provide enough privacy as they display the most important biometric features, and there are effective face reconstruction techniques [43, 48] for eigenfaces as demonstrated in Figure 4, which shows the same set of eigenfaces (available in Figure 3) after noise addition by PEEP with $\varepsilon = 4$. As the figure shows, the naked eye cannot detect any biometric features from the perturbed eigenfaces. Even at an extreme case of a privacy budget ($\varepsilon = 100$), the perturbed eigenfaces show mild levels of facial features to the naked eyes, as shown in Figure 5.

[3]http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

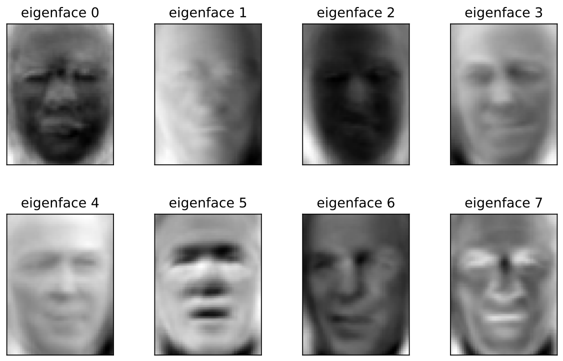

Figure 3: Eigenfaces. The figure shows a collection of sample eigenfaces generated from the input face images. The eigenfaces show only the most essential features of the input images.

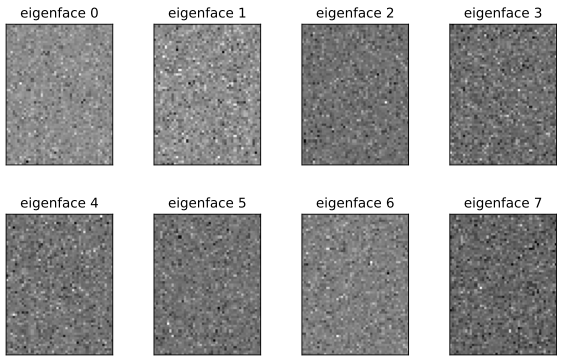

Figure 4: Perturbed eigenfaces at $\varepsilon = 4$. The randomized images appear to show no biometric features to the naked eye at $\varepsilon = 4$.

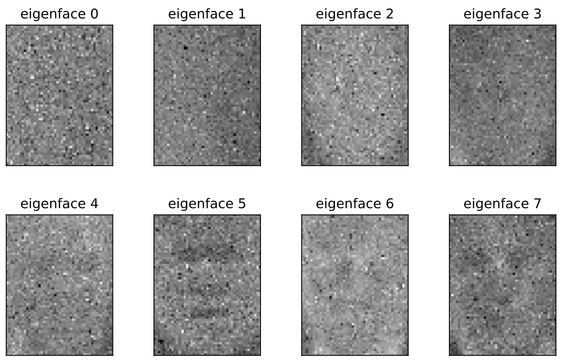

Figure 5: Perturbed eigenfaces at $\varepsilon = 100$. Here we try to demonstrate that even at an extreme case of the privacy budget (which is 100 and is not an acceptable value for ε, since $0 < \varepsilon \leq 9$ is considered as the acceptable range for ε [11]), PEEP is capable of hiding a lot of biometric features from the eigenfaces.

5. Results and Discussion

In this section, we discuss the experiments, experimental configurations, and their results. We used MLPClassifier to test the accuracy of face recognition with PEEP. MLPClassifier is a multi-layer perceptron classifier available in the scikit learn[4] Python library. We conducted all the experiments on a Windows 10 (Home 64-bit, Build 17134) computer with Intel (R) i5-6200U (6^{th} generation) CPU (2 cores with 4 logical threads, 2.3 GHz with turbo up to 2.8 GHz) and 8192 MB RAM. Then we provide an efficiency comparison and a privacy comparison of PEEP against two other privacy-preserving face recognition approaches developed by Zekeriya Erkin et al. (we abbreviate it as ZEYN for simplicity) [4] and Ahman-Reza Sadehi et al. (we abbreviate it as ANRA for simplicity) [8]. Both ZEYN and ANRA are cryptographic methods that use homomorphic encryption.

5.1. Training the MLPClassifier for perturbed eigenface recognition

We trained the MLPClassifier[5] under different levels of ε ranging from 0.5 to 8 as plotted in Figure 7. Due to the heavy noise, the datasets with lower privacy budgets exhibited difficulty for training the MLPClassifier. However, we didn't conduct any parameter tuning to increase the performance of the MLPClassifier in order to make sure that we investigate the absolute impact of perturbation on the model. Figure 6 shows the model loss of the training process of MLPClassifier when $\varepsilon = 4$. As the figure shows, the model converges after around 14 epochs.

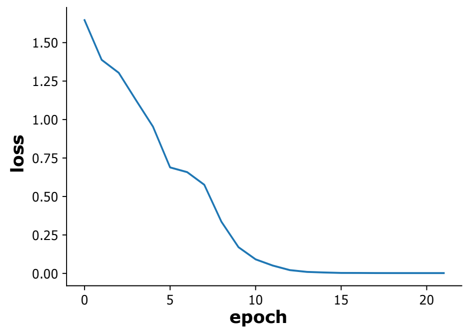

Figure 6: Model loss when PEEP with $\varepsilon = 4$. As shown in the figure, the MLPClassifier converges after around 14 epochs.

[4]https://scikit-learn.org/stable/index.html

[5]Settings used for the MLP classifier; activation='relu', batch_size=100, early_stopping =False, hidden_layer_sizes=(512, 1024, 2014, 1024, 512), max_iter =200, shuffle=True, and solver='adam', alpha=0.0001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, learning_rate='constant', learning_rate_init=0.001, momentum=0.9, nesterovs_momentum=True, power_t=0.5, random_state=None, tol=0.0001, validation_fraction=0.1, verbose=True, warm_start=False.

5.2. *Classification accuracy vs. privacy budget*

We recorded the accuracy of the trained MLPClassifier in the means of the weighted average of precision, recall, and f_1-score against varying levels of privacy budget, and plotted the corresponding data as shown in Figure 7. As discussed in Section 4.8, the class, "George W Bush" showed a higher performance as there was a higher proportion of the input image instances related to that class. As shown in Figure 7, increasing the privacy budget increases accuracy, as higher privacy budgets impose less amount of randomization on the eigenfaces. We can see that PEEP produces reasonable accuracy for privacy budgets greater than 4 and less than or equal to 8, where $0 < \varepsilon \leq 9$ is considered as an acceptable level of privacy [11].

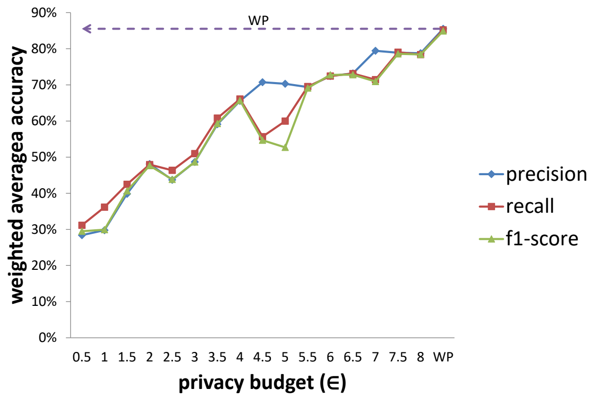

Figure 7: Performance of face recognition with privacy introduced by PEEP. WP refers to the instance of classification model without privacy where no randomization is applied to the input images.

Figure 8 shows the classification results of 8 random input images in the testing sample at $\varepsilon = 4$. According to the figure, only in one case out of eight have been misclassified. The parameters such as the minimum number of faces per each class, the size of the input dataset, and the hyperparameters of the MLPClassifier have a direct impact on accuracy. We can improve the accuracy of the MLPClassifier by changing the input parameters and conducting hyperparameter tuning. Moreover, the dataset has a higher number of instances for the class "George W Bush" compared to the other classes. A more balanced dataset would also provide better accuracy. However, in this paper, we investigate only the absolute effect of the privacy parameters on the performance of the MLPClassifier.

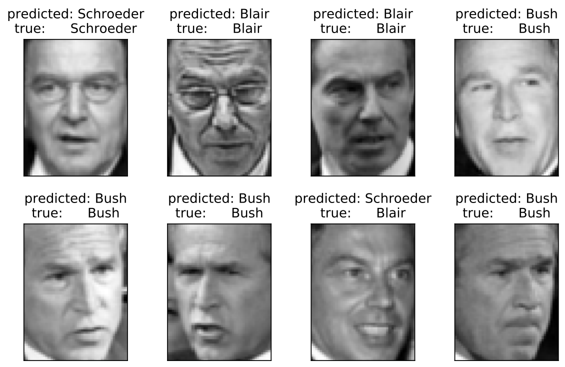

Figure 8: Instance of the face recognition when the images are randomized using PEEP at $\varepsilon = 4$ (the randomized images at $\varepsilon = 4$ are shown in Figure 4). The figure shows the predicted labels of the images against the original true labels.

5.3. Effect of *imthresh* on the performance of face recognition

In this section, we test the effect of *imthresh* (the number of images per single face) on the performance of face recognition (refer to Figure 9). During the experiment, we maintained an ε value of 8 and the number of PCA components at 128. As shown in the plots, the performance of classification improves with *imthresh*. This is a predicted observation as face recognition is a classification problem. A higher value of *imthresh* provides a higher representation for the corresponding face (class), generating higher accuracy. Hence, the proposed concept prefers a higher value for *imthresh*. This feature encourages having the highest value possible for *imthresh*, in order to generate the highest accuracy possible.

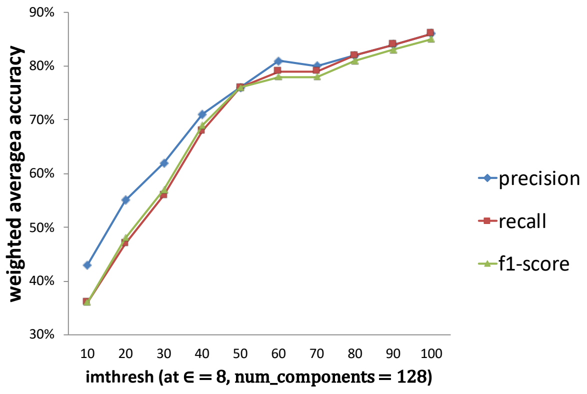

Figure 9: Performance of face recognition Vs. *imthresh*.

5.4. Effect of the number of PCA components on the performance of face recognition

In this section, we investigate the effect of the number of PCA components on the performance of face recognition. During the experiment, we maintained an ε value of 8, and *imthresh* was maintained at 100. As shown by the plot (refer Figure 10), there is an immediate increment of performance when the number of PCA components increased from 10 to 20. As the number of PCA components increase, there is a gradual increase in performance after 20 PCA components. This is due to the first 20 to 40 PCA components representing the most significant features of the input images. Although the effect of the number of PCA components after 40 is low, the improved performance suggests that it is better to have a higher number of PCA components to produce better performance.

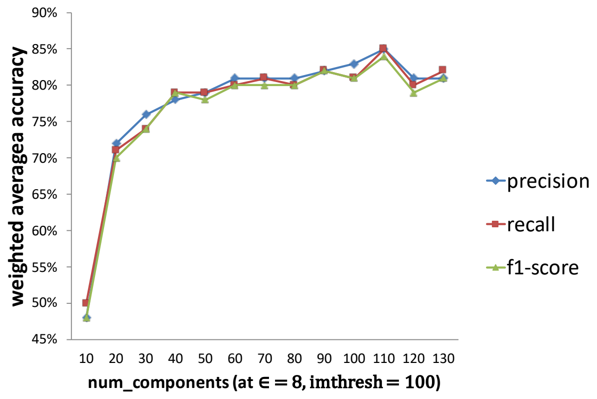

Figure 10: Performance of face recognition Vs. the number of PCA components.

5.5. Face reconstruction attack setup

It is essential that the randomized images cannot be used to reconstruct the original images that reveal the owners' identities. We prepared an experimental setup to investigate the robustness of PEEP against face reconstruction [43, 48] applied by adversaries on the randomized images.

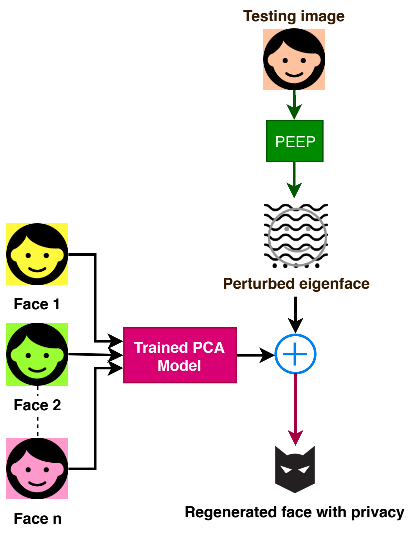

Figure 11: Face reconstruction from perturbed eigenfaces. The figure shows the experimental setup used for the reconstruction of the original input face images using the perturbed eigenfaces.

As shown in Figure 11, first, we create a PCAmodel (PCA: Principal Component Analysis) using 2,000 training images (first 1,000 images of the CelebA database and the vertically flipped versions of them). The resolution of each image is 89×109. The trained PCAmodel has the 2,000 eigenvectors of length 29,103 ($89 \times 109 \times 3$), and the mean vector (of 2,000 eigenvectors) of length 29,103. Next, the testing image (of size $89 \times 109 \times$) is read and flattened to form a vectorized form of the original image. The mean vector is then subtracted from it, and the resulting vector is randomized using PEEP to generate the privacy-preserving representation of the testing vector ($\mathcal{PV}$). Finally, we generate the eigenfaces ($\mathcal{F}_i$) and the average face by reshaping the eigenvectors ($\mathcal{FV}_i$) and mean vector available in the PCAmodel. Now we can reconstruct the original testing image from $\mathcal{PV}$ using Equation 14 where n is the number of training images used for the PCAmodel, and $\mathcal{RI}$ is the recovered image.

$$\mathcal{RI} = \sum_{i=1}^{n} \mathcal{F}_i \times (\mathcal{PV} \bullet \mathcal{FV}_i) \tag{14}$$

5.6. Empirical privacy of PEEP

Figure 12 shows the effectiveness of eigenface reconstruction attack (explained in Section 5.5) of a face image. The figure includes the results of the attack on two testing images. Figure 4 provides the empirical evidence to the level of privacy rendered by PEEP in which the lower the ε, the higher the privacy. At $\varepsilon = 0.5$, the attack is not successful in generating any underlying features of an image. At $\varepsilon = 4$ and above, we can see that the reconstructed images have some features, but they are not detailed enough to identify the person shown in that image.

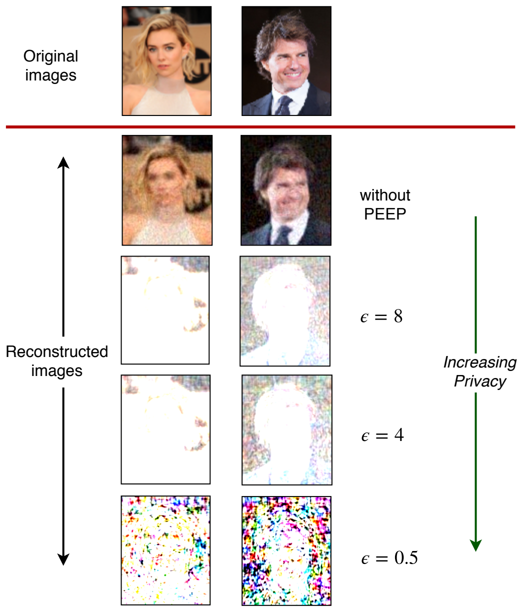

Figure 12: Reconstructing images using the setup depicted in Figure 11. The first row shows original images. The second row shows the reconstructed images using the eigenfaces of the first row images without privacy. The three remaining rows show the face reconstruction at the privacy levels of ε equals to $8, 4$, and 0.5, respectively.

5.7. Performance of PEEP against other approaches

In this section, we discuss the privacy guarantee of PEEP and the comparable methods with regards to five privacy issues (TYIS 1, 2, 3, 4, and 5) in face recognition systems, as identified in Section 1. The first six rows of Table 1 provide the summary of the evaluation, where a tick mark indicates effective addressing of a particular issue, while a cross mark shows failure. Partially addressed issues are denoted by a "∂" symbol. PEEP satisfies TYIS 1 and TYIS 4 by randomizing the input images (both training and testing) so that the randomized images do not provide any linkability to other sensitive data. Both ZEYN and ANRA are semi-honest mechanisms and need database owners to maintain the facial image databases. ZEYN and ANRA satisfy TYIS 1, if and only if the database owners are fully trusted, which can be challenging in a cloud setting, as untrusted third parties with malicious intent can access the cloud servers. As shown in Section 5.6, the randomized eigenfaces cannot be used to reconstruct original images. As the PEEP stores only randomized data in the servers, PEEP does not have to worry about the security of the cloud server. As a result, any data leak from the cloud server will not have an adverse effect on user privacy. The scalability results of the three methods given in the last row of Table 1 show that PEEP satisfies TYIS 2 by providing better scalability than ZEYN and ANRA. PEEP satisfies TYIS 3 because it uses no trusted party, whereas ZEYN and ANRA must have trusted database owners. PEEP provides some level of guarantee towards TYIS 5 by randomizing all the subsequent face image inputs related to the same person, which can come from the same device or different devices. Consequently, two input images related to the same person will have two different levels of randomization, leaving a low probability of linkability.

Table 1: Performance of PEEP against other approaches

Qualitative comparison	Type of issue (TYIS)	ZEYN	ANRA	PEEP
	1. biometric should not be linkable to other sensitive data	∂	∂	✓
	2. scalable and resource friendly	✗	✗	✓
	3. biometrics should not be accessible by a third-party	✗	✗	✓
	4. biometrics of the same person from two applications should not be linkable	∂	∂	✓
	5. biometrics should be revocable	✗	✗	∂
Quantitative comparison	Average time to recognize one image in seconds when the database has 798 images	$\sim$ 24 to 43	$\sim$ 10	0.006

✓ = fully satisfied, ∂ = partially satisfied, ✗ = not satisfied

5.8. *Computational complexity*

PEEP involves two independent segments (components) in recognizing a particular face image. Component 1 is the randomization process, and component 2 is the recognition process. The two components conduct independent operations; hence they need independent evaluations for computational complexity. Moreover, as PEEP does not need a secure communication channel, the complexity behind maintaining a secure channel does not affect the performance of PEEP. For a particular instance of PEEP (refer to Algorithm 3), step 11 to step 12 display linear complexity of $O(nc)$, where nc is the number of principal components, and the image resolution (width in pixels, height in pixels) will remain constant during a particular instance of perturbation and recognition. When width in pixels=47, height in pixels=62, and the number of PCA components=128, PEEP takes around 0.004 seconds to randomize a single input image. Component 2 can be composed of any suitable classification model; in our case, we use the MLPClassifier (refer Section 5.1) as the facial recognition module that was trained using 798 images. Under the same input settings (width in pixels=47, height in pixels=62, and the number of PCA components=128), the trained model takes 0.002 seconds to recognize a facial image input. Since the prediction is always done on a converged model, the time taken for prediction will be constant and follow a complexity of $O(1)$. For randomization and prediction PEEP roughly consumes around 0.006 seconds under the given experimental settings. The runtime plots shown in Figure 13 further validate the computational complexities evaluated above. According to the last row of Table 1, PEEP is considerably faster than comparative methods; PEEP provides a more effective and efficient approach towards the recognition of images against millions of faces in a privacy-preserving manner. In further examining the performance of PEEP for its efficiency, we investigated PE-MIU [20], and POR [21] (refer to Section 2), which are two recently developed approaches. PE-MIU consumes a complete MIU-verification time of 0.0072 seconds for a block size of 4 in a computer with an Intel(R) Core(TM) i7-7700 processor. POR consumes a testing time of around 0.011 seconds per one image in an Intel(R) Core(TM) i5-7200 CPU @2.50GHz and 8.00GB of RAM. Hence, under the proposed experimental settings, a prediction time of 0.006 seconds consumed by PEEP can be considered as efficient and reliable.

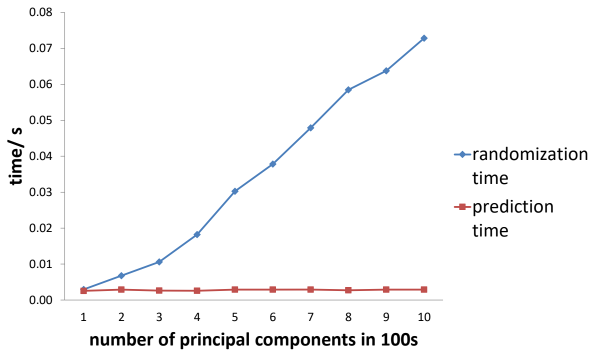

Figure 13: The time consumption of PEEP to randomize and recognize one input image against the increasing number of principal components used for the eigenface generation.

6. Conclusions

We proposed a novel mechanism named PEEP for privacy-preserving face recognition using data perturbation. PEEP utilizes the properties of differential privacy, which can provide a strong level of privacy to facial recognition technologies. PEEP does not need a trusted party and employs a local approach where randomization is applied before the images reach an untrusted server. PEEP forwards only randomized data, which requires no secure channel. PEEP is an efficient and lightweight approach that can be easily integrated into any resource-constrained device. As the training and testing/recognition of facial images done solely on the randomized data, PEEP does not incur any efficiency loss during the recognition of a face. The differentially private notions allow users to tweak the privacy parameters according to domain requirements. All things considered, PEEP is a state of the art approach for privacy-preserving face recognition.

Using the proposed approach with different biometric algorithms and areas like fingerprint and iris recognition will be looked at in the future, in particular with regards to effectiveness and sensitivity in different domains of inputs.

References

[1] B. Mandal, S.-C. Chia, L. Li, V. Chandrasekhar, C. Tan, and J.-H. Lim, "A wearable face recognition system on google glass for assisting social interactions," in *Asian Conference on Computer Vision*, pp. 419–433, Springer, 2014.

[2] M. MacAulay and M. D. Moldes, "Queen don't compute: reading and casting shade on facebook's real names policy," *Critical Studies in Media Communication*, vol. 33, no. 1, pp. 6–22, 2016.

[3] R. Bhagavatula, B. Ur, K. Iacovino, S. M. Kywe, L. F. Cranor, and M. Savvides, "Biometric authentication on iphone and android: Usability, perceptions, and influences on adoption," in *USEC '15*, Internet Society, 2015.

[4] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *International Symposium on Privacy Enhancing Technologies Symposium*, pp. 235–253, Springer, 2009.

[5] M. A. P. Chamikara, P. Bertok, D. Liu, S. Camtepe, and I. Khalil, "An efficient and scalable privacy preserving algorithm for big data and data streams," *Computers & Security*, vol. 87, p. 101570, 2019.

[6] M. A. P. Chamikara, P. Bertok, D. Liu, S. Camtepe, and I. Khalil, "Efficient data perturbation for privacy preserving and accurate data stream mining," *Pervasive and Mobile Computing*, vol. 48, pp. 1–19, 2018.

[7] M. A. P. Chamikara, A. Galappaththi, R. D. Yapa, R. D. Nawarathna, S. R. Kodituwakku, J. Gunatilake, A. A. C. A. Jayathilake, and L. Liyanage, "Fuzzy based binary feature profiling for modus operandi analysis," *PeerJ Computer Science*, vol. 2, p. e65, 2016.

[8] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *International Conference on Information Security and Cryptology*, pp. 229–244, Springer, 2009.

[9] C. Xiang, C. Tang, Y. Cai, and Q. Xu, "Privacy-preserving face recognition with outsourced computation," *Soft Computing*, vol. 20, no. 9, pp. 3735–3744, 2016.

[10] E. M. Newton, L. Sweeney, and B. Malin, "Preserving privacy by de-identifying face images," *IEEE transactions on Knowledge and Data Engineering*, vol. 17, no. 2, pp. 232–243, 2005.

[11] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, ACM, 2016.

[12] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman, "Local differential privacy for deep learning," *IEEE Internet of Things Journal*, 2019.

[13] T. Heseltine, N. Pears, J. Austin, and Z. Chen, "Face recognition: A comparison of appearance-based approaches," in *Proc. VIIth Digital image computing: Techniques and applications*, vol. 1, 2003.

[14] F. Tsalakanidou, D. Tzovaras, and M. G. Strintzis, "Use of depth and colour eigenfaces for face recognition," *Pattern recognition letters*, vol. 24, no. 9-10, pp. 1427–1435, 2003.

[15] K. Delac, M. Grgic, and P. Liatsis, "Appearance-based statistical methods for face recognition," in *47th International Symposium ELMAR-2005*, pp. 151–158, 2005.

[16] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," 2015.

[17] R. Cendrillon and B. Lovell, "Real-time face recognition using eigenfaces," in *Visual Communications and Image Processing 2000*, vol. 4067, pp. 269–276, International Society for Optics and Photonics, 2000.

[18] A. Bhargav-Spantzel, A. C. Squicciarini, S. Modi, M. Young, E. Bertino, and S. J. Elliott, "Privacy preserving multi-factor authentication with biometrics," *Journal of Computer Security*, vol. 15, no. 5, pp. 529–560, 2007.

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

[20] P. Terhörst, K. Riehl, N. Damer, P. Rot, B. Bortolato, F. Kirchbuchner, V. Struc, and A. Kuijper, "Pe-miu: A training-free privacy-enhancing face recognition approach based on minimum information units," *IEEE Access*, 2020.

[21] Z. Ma, Y. Liu, X. Liu, J. Ma, and K. Ren, "Lightweight privacy-preserving ensemble classification for face recognition," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5778–5790, 2019.

[22] M. A. P. Chamikara, P. Bertok, D. Liu, S. Camtepe, and I. Khalil, "Efficient privacy preservation of big data for accurate data mining," *Information Sciences*, 2019.

[23] F. Dufaux and T. Ebrahimi, "Scrambling for video surveillance with privacy," in *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pp. 160–160, IEEE, 2006.

[24] X. Yu, K. Chinomi, T. Koshimizu, N. Nitta, Y. Ito, and N. Babaguchi, "Privacy protecting visual processing for secure video surveillance," in *2008 15th IEEE International Conference on Image Processing*, pp. 1672–1675, IEEE, 2008.

[25] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, *et al.*, "Privacy-preserving fingercode authentication," in *Proceedings of the 12th ACM workshop on Multimedia and security*, pp. 231–240, ACM, 2010.

[26] C. Rathgeb and A. Uhl, "Privacy preserving key generation for iris biometrics," in *IFIP International Conference on Communications and Multimedia Security*, pp. 191–200, Springer, 2010.

[27] K. Gai, M. Qiu, H. Zhao, and J. Xiong, "Privacy-aware adaptive data encryption strategy of big data in cloud computing," in *Cyber Security and Cloud Computing (CSCloud), 2016 IEEE 3rd International Conference on*, pp. 273–278, IEEE, 2016.

[28] M. J. Brady, "Biometric recognition using a classification neural network," Apr. 6 1999. US Patent 5,892,838.

[29] K. Yang, Q. Han, H. Li, K. Zheng, Z. Su, and X. Shen, "An efficient and fine-grained big data access control scheme with privacy-preserving policy," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 563–571, 2017.

[30] J. Zhong, V. Mirchandani, P. Bertok, and J. Harland, "μ-fractal based data perturbation algorithm for privacy protection.," in *PACIS*, p. 148, 2012.

[31] A. Machanavajjhala and D. Kifer, "Designing statistical privacy for your data," *Communications of the ACM*, vol. 58, no. 3, pp. 58–67, 2015.

[32] C. Dwork, "The differential privacy frontier," in *Theory of Cryptography Conference*, pp. 496–502, Springer, 2009.

[33] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman, "A trustworthy privacy preserving framework for machine learning in industrial iot systems," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[34] T. Chanyaswad, A. Dytso, H. V. Poor, and P. Mittal, "Mvg mechanism: Differential privacy under matrix-valued query," *arXiv preprint arXiv:1801.00823*, 2018.

[35] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren, "Heavy hitter estimation over set-valued data with local differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 192–203, ACM, 2016.

[36] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *Advances in neural information processing systems*, pp. 2879–2887, 2014.

[37] F. D. McSherry, "Privacy integrated queries: an extensible platform for privacy-preserving data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 19–30, ACM, 2009.

[38] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 1054–1067, ACM, 2014.

[39] T.-H. H. Chan, M. Li, E. Shi, and W. Xu, "Differentially private continual monitoring of heavy hitters from distributed streams," in *International Symposium on Privacy Enhancing Technologies Symposium*, pp. 140–159, Springer, 2012.

[40] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[41] Y. Wang, X. Wu, and D. Hu, "Using randomized response for differential privacy preserving data collection.," in *EDBT/ICDT Workshops*, vol. 1558, 2016.

[42] J. Zhang, Y. Yan, and M. Lades, "Face recognition: eigenface, elastic matching, and neural nets," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1423–1435, 1997.

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

[44] J. Bennett, R. Grout, P. Pébay, D. Roe, and D. Thompson, "Numerically stable, single-pass, parallel statistics algorithms," in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pp. 1–8, IEEE, 2009.

[45] M. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe, "Privacy preserving distributed machine learning with federated learning," *arXiv preprint arXiv:2004.12108*, 2020.

[46] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Theory of Cryptography Conference*, pp. 635–658, Springer, 2016.

[47] C. C. Aggarwal and P. S. Yu, "A condensation approach to privacy preserving data mining," in *EDBT*, vol. 4, pp. 183–199, Springer, 2004.

[48] D. Pissarenko, "Eigenface-based facial recognition," *December 1st*, 2002.