Computers and Electrical Engineering 89 (2021) 106924

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Secure and verifiable iris authentication system using fully homomorphic encryption

Mahesh Kumar Morampudi [a,b,*], Munaga V.N.K. Prasad [a], Mridula Verma [a], U.S.N. Raju [b]

[a] Center for Affordable Technologies, Institute for Development and Research in Banking Technology (IDRBT), Hyderabad, India
[b] Department of Computer Science and Engineering, National Institute of Technology, Warangal, India

ARTICLE INFO

Keywords:
Biometrics
Privacy-preserving
Homomorphic encryption
Multi-class Perceptron
Nearest Neighbor

ABSTRACT

With the escalated usage of a biometric authentication system (BAS), template protection for biometrics attracted research interest in recent years. The assumption behind the existing homomorphic encryption-based BASs is that the server performs the computations honestly. In a malicious server setting, the server may return an arbitrary result to save the computational resources, which may result in false accept/reject. To tackle this challenge, we propose a secure and verifiable classification based iris authentication system (SvaS). SvaS aims to achieve both privacy-preserving (PP) training and PP classification of Nearest Neighbor and Multi-class Perceptron models. The Fan-vercauteren scheme provides confidentiality for the iris templates, and aggregate verification vector helps to verify the correctness of the computed classification result. Extensive experimental results on benchmark iris databases demonstrate that SvaS provides privacy to the iris templates with no loss in accuracy and eliminates the need to trust the server.

1. Introduction

Unlike, password or token authentication systems, a biometric authentication system (BAS) is more flexible for users since they do not need to carry or remember anything. Fingerprint, iris, face etc. are the commonly used biometric modalities. Biometric modalities are used to provide security on the Internet of Medical Things based remote health-care systems [1]. The properties like stability and uniqueness make the iris to be mostly used in various applications when compared to other biometric modalities [2].

The iris recognition system includes two phases, namely the enrollment and identification/verification phases. The iris template obtained from a reference iris image is stored in a centralized server during the enrollment phase. In authentication phase, the iris template obtained from the probe iris image is compared with the iris templates stored during enrollment phase. As the biometric data is unique to a person, it is irrevocable if it gets compromised. Studies such as [3], proved that an iris image could be reconstructed from its template. The unauthorized access to biometric templates results several attacks like hill-climbing attack, replay attack, masquerade attack, and stolen-token attack. For instance, in 2018, hackers stole 1 billion Indian user's biometrics of Aadhaar. Homomorphic Encryption (HE) is used as a template protection scheme to ensure the privacy & security to iris templates [4]. HE is categorized into Partial HE (PHE), Somewhat HE (SHE), and Fully HE (FHE). FHE offers to perform unlimited multiplications & additions on the encrypted data. SHE offers to perform a limited number of addition & multiplication on the encrypted data. PHE offers to perform either multiplication or addition but not both on the encrypted data.

* Corresponding author at: Center for Affordable Technologies, Institute for Development and Research in Banking Technology (IDRBT), Hyderabad, India.
E-mail address: morampudimahesh@gmail.com (M.K. Morampudi).

https://doi.org/10.1016/j.compeleceng.2020.106924
Received 30 December 2019; Received in revised form 6 August 2020; Accepted 18 November 2020
Available online 23 November 2020

The early secure biometric authentication system was designed in an outsourced environment by Sedenka et al. [5]. Haghighat et al. [6] suggested a biometric verification in a cloud environment. The method used a searching-based matching instead of distance-based matching. Xiang et al. [7] introduced a secure face recognition with computation in a cloud server by using public key encryption & fully homomorphic encryption algorithm. Gomez et al. [8] discussed a template protection approach for multi-biometric recognition using Paillier method. The final comparison was performed on the unencrypted data by the server; as a result, introduced a breach into the security of the system. A light-weighted encryption scheme named "Threshold Predicate Encryption (TPE)" was proposed by Zhou et al. [9]. A privacy-preserving user-centric biometric authentication system (PassBio) was proposed by using TPE. Barni et al. [10] introduced a secure multi-modal biometric authentication ("SEMBA"), which combines face & iris templates. The above methods assume that the server is "Honest-but-curious".

Machine learning techniques have been providing useful tools for solving a wide range of problems in multiple domains including computer vision, bio-informatics, biometric recognition and many more. Since last decade, machine learning algorithms are used to detect, segment and recognize the irises effectively. Rai et al. [11] suggested a method to identify the iris patterns by using support vector machines (SVM) and Hamming distance. Authors proposed two feature extraction techniques, namely 1D Log Gabor wavelet and Haar wavelet decomposition. An iris recognition system was proposed by Ahmadi et al. [12] to increase generalization performance by using particle swarm optimization & multi-layer perceptron (MLP). The authors extended their work in [13] by using radial basis function (RBF) with a genetic algorithm to reduce the computational complexity. A deep learning model was designed by Arsalan et al. [14], to determine the exact iris region without pre-processing the eye image. Unlike existing approaches, the performance is not affected by non-ideal situations. Zhao & Ajay [15] used fully convolutional network and proposed a framework for accurate iris detection, segmentation and recognition. Authors developed an "Extended Triplet Loss (ETL)" function to learn the spatially corresponding features of an iris image.

The emergence of machine learning as a service originates the privacy dilemma. According to this dilemma, either the model is known to the user or the user's private data is revealed to the entity, which evaluates the machine-learning model. Training and testing are the two phases involved in machine learning classification (MLC) task. The algorithm learns a model w using a set of labeled instances in the training phase. A classifier C is run over a new feature vector x, using the model w to output a prediction $C(x, w)$. Private machine learning classification (PMLC) is a method, in which both training & testing takes place on encrypted data. FHE, which supports homomorphic operations on encrypted data without decryption, contributes to MLC without leaking user privacy, especially in the outsourcing scenario. Recently, various PMLC schemes have been constructed [16,17] which can perform MLC on encrypted data but fails to attain both Privacy-preserving (PP) training as well as PP classification at the same time.

Most of the existing BASs based on HE assume a "Honest-but-Curious" server to provide privacy for biometric templates. Therefore, the existing methods only solve the modify templates attack of BAS and fail to overcome the override comparator attack of BAS. On the other hand, most of the existing MLC methods on encrypted data provides either PP training or PP classification but not both. In BAS, it is essential to conduct training & testing on encrypted iris templates to provide the privacy of both iris templates and the model. Therefore, in this paper we construct a method which do not compromise privacy of templates and provides privacy and trust to the classification result. The main contributions of our method are listed below:

- We propose a secure and verifiable machine learning-based iris authentication system (SvaS), which performs PP training, PP classification and eliminates the need to trust the centralized server.
- We propose private nearest neighbor (PNN) and private multi-class perceptron (PMCP) algorithms to perform both training and classification phases on encrypted data.
- SvaS includes a verification procedure to verify whether the classification result computed by the cloud server is correct or not. The verification procedure allows a public verifier to validate the result without using the private information of the user.
- SvaS solves the attacks like modify templates, intercept channel and override comparator in BAS.
- We show the effectiveness of SvaS by experimenting it on publicly available benchmark iris databases.

Organization The rest of the paper is organized as follows. Section 2 presents a system to authenticate the user using PNN and PMCP. Section 3 illustrates the implementation details and experimental results analysis. The conclusion and future scope are given in Section 4.

2. Secure and verifiable machine learning based iris authentication system using FHE

To the best of our knowledge, SvaS is the first iris authentication system which provides confidentiality of the iris template as well as trust on the computed result. Table 1 presents the list of notations used in SvaS. Fig. 1 shows the flow diagram of SvaS. The client device, cloud server, authentication server, and public verifier are the four entities involved in SvaS. The role of authentication server is to (1) Generate secret (δ_d) and public (δ_e) keys. (2) Send accept/reject decision to the client device. The cloud server provides the classification service & storage to the client device. During the training phase, the cloud server builds a machine learning model. The end-user is authenticated using the generated model in the testing phase. The false accept/reject may happen if the cloud server does not perform the computations honestly. So, the correctness of the classification result computed by the cloud server is verified by the public verifier to avoid false acceptances/rejections. Instead of returning the class label during the classification phase, PNN classifier returns the Manhattan distances, and PMCP returns the dot product values to the public verifier. Algorithm 1 and Algorithm 2 depicts the steps involved in the enrollment & authentication phases of SvaS.

Table 1
Notations used in SvaS.

Parameter	Description
a, n, q	Modulus in the plaintext space, A power of 2, Modulus in the ciphertext space.
$x^n + 1$	The polynomial modulus which specifies the ring R.
R	The ring $\mathbb{Z}[x]/(x^n + 1)$.
R_a	The ring $\mathbb{Z}_a[x]/(x^n + 1)$ i.e., same as the ring R but with coefficient reduced modulo a.
ω	A base into which cipher text elements are decomposed during relinearization.
d	Dimensions of the feature vector i.e., iris template.
N	Number of reference iris templates i.e., training instances.
c	Number of classes i.e., subjects.
δ_e, δ_d	Public & secret keys.
$\{X_i\}_{i=1}^N$	Reference iris template with d dimension.
Y	Probe iris template.
$\{w_i\}_{i=1}^c$	Weight vectors.
$\text{Enc}(\delta_e, X)$	The encryption of X with δ_e.
$\epsilon(a)$	Encrypted value of a.
$\epsilon(Z_{n+1})$	Encrypted verification vector generated by authentication server. The dimension of $\epsilon(Z_{n+1})$ is same as reference template.
$\epsilon(v_i)$	Encrypted random integer.
$\epsilon(R)$	Computed Manhattan distances in the case of *nearest neighbor* (NN) and dot products in the case of *multi-class perceptron* (MCP).
id	Identifier of the end-user.
l	Class label of the training instance.

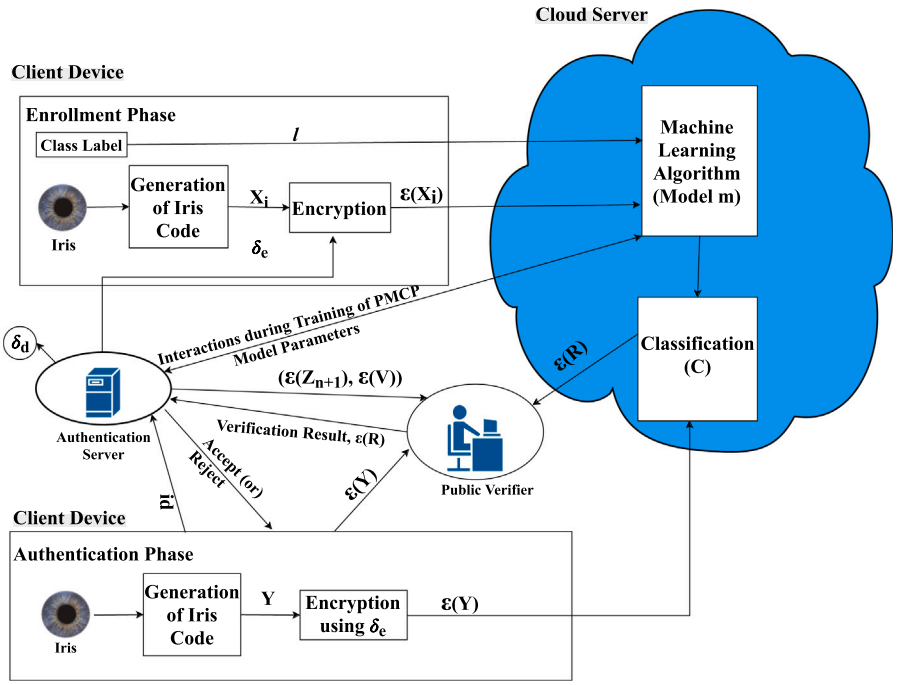


Fig. 1. Block diagram of SvaS. The model is only accessible to the server and the input i.e., iris code is accessible only to the client in both the training and testing phases.

2.1. Assumptions

SvaS assume the following

1. During the enrollment and authentication phases, the client device is a trusted entity and has limited computational and memory resources.
2. The cloud server does not perform the computations honestly.
3. The authentication server is a trusted entity which generates the secret & public keys differently for each user. The secret key of the user is stored securely and the public keys are broadcast to the client device.
4. The public verifier is only trusted to check the correctness of $\epsilon(R)$.

Algorithm 1 Enrollment Phase

Input: Reference iris image, Corresponding class label l

1: The authentication server generates the public key, δ_e and secret key, δ_d.
2: The client device generates the reference iris template, X_i from the reference iris image using the University of Salzburg Iris Toolkit [18].
3: The client device encrypts X_i and sends $\varepsilon(X_i)$ along with a class label to the cloud server.
4: The cloud server applies PP training on encrypted reference iris templates $\varepsilon(X_i)$, $i \in [1, N]$ using PMCP or PNN and generates a model.
5: The cloud server sends the model parameters of PMCP, $\varepsilon(w[f])$, $f \in [1, c]$ i.e., weight vectors and parameters of PNN, $\varepsilon(X_i)$, $i \in [1, N]$ to authentication server.
6: The authentication server generates the encrypted verification vector, $\varepsilon(Z_{n+1})$, encrypted random vector, $\varepsilon(V)$ separately for each classifier using the model parameters.

Algorithm 2 Authentication Phase

Input: Probe iris image, Identifier or class label id of the end user
Output: Reject or Accept

1: The authentication server send the $\varepsilon(Z_{n+1})$ & $\varepsilon(V)$ to the public verifier.
2: The client device generates the probe iris template, Y from the probe iris image using the University of Salzburg Iris Toolkit [18]. It also acquires the identifier id of the end-user and sends id to the authentication server.
3: The client device encrypts Y and sends $\varepsilon(Y)$ to the cloud server.
4: The cloud server calculate the classification result, $\varepsilon(R)$ and send to the public verifier. (Instead of returning the class label, PNN returns the encrypted Manhattan distance between $\varepsilon(Y)$ and $\varepsilon(X_i)$ and PMCP returns the dot products between $\varepsilon(Y)$ and $\varepsilon(w[i])$, $i \in [1, c]$).
5: The public verifier checks the correctness of the computed result $\varepsilon(R)$ by using $\varepsilon(Z_{n+1})$, $\varepsilon(V)$, $\varepsilon(Y)$ and sends the verification result to authentication server.
6: If the verification succeeds, then the authentication server computes the predicted class label and compares with id given by the end-user to determine whether the user is genuine or not.

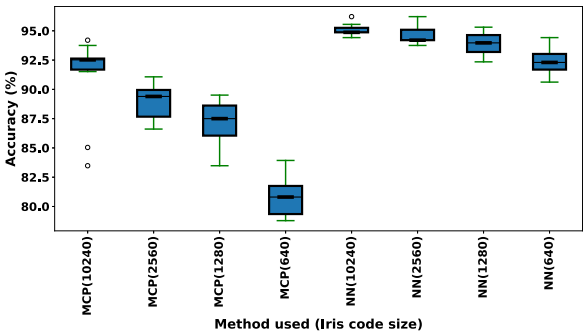


Fig. 2. Comparison of accuracy between iris code of sizes 10 240, 2560, 1280 and 640 for MCP and NN.

2.2. Generation of iris codes and encryption

Compression of iris codes and encryption are the two phases involved in this section. The compression of the iris code phase helps to reduce the size of the iris templates results in the improvement of the system performance. The compressed iris code is encoded into polynomials, and encoded polynomial is encrypted in ensuring the confidentiality of iris templates (encryption) phase.

2.2.1. Compression of the iris template

The size of the iris template determines the performance of the system. So, the 10 240-bit binary vector is grouped into blocks of size m by using Eq. (1). m denotes the size of the block, and we consider 4, 8, and 16 as m values. These m-bits are converted to decimal values and stored in a vector. Each integer value is further divided by 2 to obtain the binary vector. The 2560 integers are replaced with binary values.

$$compressed\ iris\ template\ size = \frac{Total\ no.\ of\ bits}{m} \tag{1}$$

The accuracies obtained for the original iris template and different sizes of iris template are shown in Fig. 2. From Fig. 2, it is observed that there is a slight variation of accuracy between the actual iris code and compressed iris template of size 2560 for both MCP and NN. Hence, SvaS considers the compressed iris template as a feature vector instead of the original iris template.

2.2.2. Ensuring the confidentiality of iris templates

The compressed iris templates are encoded with a polynomial ring R_a using Batching scheme [19] results in the performance improvement of homomorphic multiplication & addition. The encoded polynomial is encrypted using the Brakerski Fan–Vercuateran (BFV) scheme [19] to ensure the confidentiality of iris templates and to perform computations on the encrypted data.

2.3. Secure and verifiable machine learning classification

The procedures to implement the NN & MCP on the encrypted data is described in this section. The advantage of PNN and PMCP is that they provide privacy not only to the iris templates but also to the model by performing both PP training and PP classification. The model is only accessible to the server, and the training & test instances are only accessible to the client device.

2.3.1. Private nearest neighbor (PNN)

The aim of PNN is to achieve NN on encrypted templates. Instead of returning the class label, PNN returns the Manhattan distances between $\varepsilon(X_i)$ and $\varepsilon(Y)$.

$$R = \{r_i / r_i = \sum_{j=1}^{d} (X_i[j] - Y[j]), \forall\ i = 1\ to\ N\} \tag{2}$$

The server does not learn either $\varepsilon(X_i)$ or $\varepsilon(Y)$. In particular, we show how the server can execute Eq. (2) when both the training and testing instances are encrypted. The detailed procedure to find the NN on encrypted data is illustrated in Algorithm 3. The inputs to the PNN are the encrypted reference templates, corresponding class labels and encrypted probe template, respectively. PNN returns the Manhattan distances, $\varepsilon(R)$ between $\varepsilon(X_i)$ and $\varepsilon(Y)$ as an output which is given in Eq. (3).

$$\varepsilon(R) = \{r_i / r_i = (\varepsilon(X_i) - \varepsilon(Y)), \forall\ i = 1\ to\ N\} \tag{3}$$

Since both the reference and the probe templates are encrypted, the privacy of iris templates, i.e., user privacy is maintained.

Algorithm 3 Nearest Neighbor on Encrypted data (PNN)

Input: $\varepsilon(X_1)$, $\varepsilon(X_2)$, . . . , $\varepsilon(X_N)$, Corresponding class labels cls_1, cls_2, . . . , cls_N, $\varepsilon(Y)$
Output: $\varepsilon(R)$

1: **begin**
2: **for** $i \leftarrow 1$ to N **do**
3: $r_i \leftarrow sub(\varepsilon(X_i), \varepsilon(Y))$
4: **for** $j \leftarrow 0$ to p **do** // *where* $p = log_w^q$
5: $r_i \leftarrow r_i + k_{g^j}(r_i)$
6: **end for**
7: $r_i \leftarrow r_i$
8: **end for**
9: $\varepsilon(R) = (r_1, r_2, ..., r_N)$
10: **return** $\varepsilon(R)$
11: **end**

Let $\varepsilon(X_i)$ and $\varepsilon(Y)$ are the encrypted vectors. The aim is to achieve Eq. (3) i.e., find the Manhattan distances between $\varepsilon(X_i)$ and $\varepsilon(Y)$ without decryption. r_i is the variable to store the subtracted result of ith encrypted reference template, $\varepsilon(X_i)$ and probe template, $\varepsilon(Y)$. To improve the performance of the system, we used batching as the encoding scheme before encrypting the reference and probe templates. Hence, with the computational cost of just one operation, we can accomplish d homomorphic subtractions. The disadvantage of batching is that it is not possible to access the individual values of the encrypted vector. Hence, it restricts to compute the sum of elements after the subtract operation (Eq. (2)). This problem can be solved by using the observation made by Gentry *et al*. [20], particularly, it is likely to rotate the encrypted vectors cyclically without decryption. As a result, if the encrypted vectors are rotated cyclically and adding the encrypted vectors $p = log_w^q$ times then the first slot of the resultant vector gives the sum value. The steps (4–6) of Algorithm 3 describes the process of cyclically rotating and adding the r_i. This operation is explained with an example in Fig. 3. The ith Manhattan result is stored in r_i. The steps (3–7) of Algorithm 3 repeat for N reference templates yields N Manhattan distances which are assigned to $\varepsilon(R)$. The cloud server computes the Manhattan distance on the encrypted data; as a result, the privacy of the iris templates is achieved. If the cloud server did not perform the Manhattan distance honestly and return a random result to minimize the use of its computational resources, then the imposter may get access into the system. To overcome this limitation, the public verifier checks the correctness of the result returned by the cloud server.

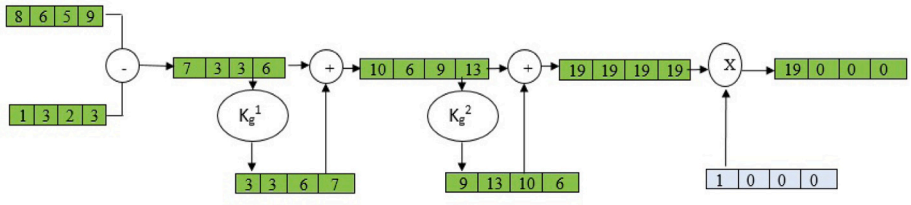


Fig. 3. Homomorphic computation of Manhattan distance between vectors when vectors are encoded using batching scheme.

2.3.2. Verification scheme for nearest neighbor

The cloud server computes the Manhattan distances $\varepsilon(R) = r_i, \forall\ i = 1\ to\ N$ between $\varepsilon(X_i)\ \forall\ i = 1\ to\ N$ and $\varepsilon(Y)$. The verification scheme allows the public verifier to check the correctness of $\varepsilon(R)$ returned by the cloud server.

Generation of encrypted verification vector: After the enrollment phase, the authentication server constructs the encrypted verification vector using the encrypted reference iris templates $\varepsilon(X_i), \forall\ i = 1\ to\ N$. The encrypted verification vector helps the public verifier to check the correctness of the Manhattan distances. Let $\varepsilon(Z_{n+1})$ be the encrypted verification vector and is defined as

$$\begin{aligned}\varepsilon(Z_{n+1}) &= (\varepsilon(X_1) - \varepsilon(v_1)) + (\varepsilon(X_2) - \varepsilon(v_2)) + \ldots . + (\varepsilon(X_N) - \varepsilon(v_N)) \\ &= \sum_{i=1}^{N} (\varepsilon(X_i) - \varepsilon(v_i))\end{aligned} \tag{4}$$

where, $v_i\ \forall\ i = 1\ to\ N$ is the random integer and $\varepsilon(V) = (\varepsilon(v_1), \varepsilon(v_2), \ldots, \varepsilon(v_N))$. As long as the secret key is secure, encrypted verification vector is also secure and its security relies on the hardness of Ring Learning With Errors (RLWE).

Ensuring the correctness of Manhattan distance: The public verifier checks the correctness of Manhattan distances, $\varepsilon(R)$ using $\varepsilon(Z_{n+1})$, $\varepsilon(Y)$ and $\varepsilon(V)$. Our verification scheme checks the correctness of the result on the encrypted data itself; as a result, anyone can perform the correctness of $\varepsilon(R)$ without the secret key. The public verifier computes $\varepsilon(D1) = (\varepsilon(Z_{n+1}) - N\varepsilon(Y))$ and $\varepsilon(D2) = \sum_{i=1}^{N} (r_i - \varepsilon(v_i))$. Finally, compute $(\varepsilon(D1) - \varepsilon(D2))$. If the result is zero, the Manhattan distances $\varepsilon(R)$ returned by the cloud server is considered to be correct. The below proof uses Eqs. (3), (4) and some algebraic properties of vectors and explains how $\varepsilon(D1)$ and $\varepsilon(D2)$ are same. If the verification succeeds then the Manhattan distances $\varepsilon(R)$ are considered to be correct. So, the authentication server finds the predicted class by computing the index of the minimum value among $\varepsilon(R)$. The computed predicted class is compared with id given by the end-user to determine whether the user is genuine or not.

$$\begin{aligned}\varepsilon(D1) &= (\varepsilon(Z_{n+1}) - N\varepsilon(Y)) \\ &= \sum_{i=1}^{N} (\varepsilon(X_i) - \varepsilon(v_i)) - N\varepsilon(Y) \\ &= \sum_{i=1}^{N} \varepsilon(X_i) - \sum_{i=1}^{N} \varepsilon(v_i) - N\varepsilon(Y) \\ &= \sum_{i=1}^{N} \varepsilon(X_i) - N\varepsilon(Y) - \sum_{i=1}^{N} \varepsilon(v_i) \\ &= \sum_{i=1}^{N} \varepsilon(X_i) - \sum_{i=1}^{N} \varepsilon(Y) - \sum_{i=1}^{N} \varepsilon(v_i) \\ &= \sum_{i=1}^{N} (\varepsilon(X_i) - \varepsilon(Y)) - N\varepsilon(v_i) \\ &= \sum_{i=1}^{N} r_i - \sum_{i=1}^{N} \varepsilon(v_i) \\ &= \sum_{i=1}^{N} (r_i - \varepsilon(v_i)) = \varepsilon(D2)\end{aligned}$$

2.3.3. Private multi-class perceptron (PMCP)

The aim of PMCP is to achieve MCP on encrypted templates. Instead of returning the class label, PMCP returns the dot products between $\varepsilon(w_i), \forall\ i = 1\ to\ c$ and $\varepsilon(Y)$.

$$\varepsilon(R) = \{r_i / r_i = \varepsilon(Y).\varepsilon(w[i]), \forall\ i = 1\ to\ c\} \tag{5}$$

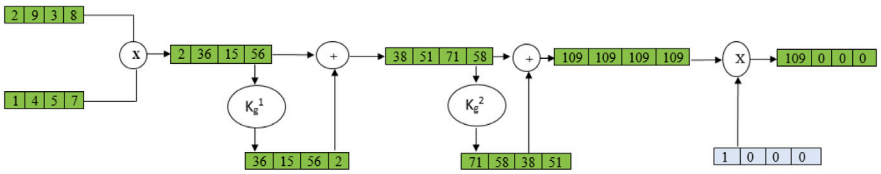


Fig. 4. Homomorphic computation of dot product between vectors when vectors are encoded with batching scheme.

Training (PMCP:training) and classification (PMCP:classification) are the two phases involved in PMCP. Algorithm 4 describes the training phase of MCP on encrypted templates i.e., PMCP:training. The PMCP:training takes the class labels of iris templates, encrypted reference iris templates, iterations T (not encrypted) and Bias as inputs and returns the encrypted weight vectors as an output. The process during training phase is explained below. Let $\varepsilon(X_1)$, $\varepsilon(X_2)$, ..., $\varepsilon(X_N)$ are the encrypted reference templates. The PMCP requires multiple training iterations to fully learn the model. During each iteration, the jth encrypted reference template is multiplied with each unique weight vector and stores in ct. As explained in Section 2.3.1, the problem with batching occurs here as well while performing the sum of the elements after the multiplication. The problem can be solved by the process of cyclically rotated and adding the encrypted vectors [20]. The steps (20–22) of Algorithm 4 describes the process of cyclically rotating and adding the ct. The element in the first slot is the desired product. This operation is explained with an example in Fig. 4. The class of the jth encrypted template is the class that gives the highest product result. If the calculated class, p_cls and the actual class, cls_j of the jth encrypted reference template are not equal then the weight vector is updated as follows: feature vector, $\varepsilon(X_j)$ is added to the actual weight vector, $\varepsilon(w[cls_j])$ and subtracted from the predicted weight vector, $\varepsilon(w[p\_cls_j])$. After the final iteration, the final encrypted weight vectors should be stable.

The detailed procedure to achieve $\varepsilon(R)$ is given in Algorithm 6. The inputs to the PMCP:classification are the encrypted probe template $\varepsilon(Y)$ and encrypted weight vectors $\varepsilon(w[i]), \forall\ i = 1\ to\ c$ respectively. PMCP:classification returns the dot product $\varepsilon(R)$ between weight vectors and $\varepsilon(Y)$ as an output which is given in Eq. (5). In Algorithm 6, r_i stores the multiplication result of $\varepsilon(Y)$ and $\varepsilon(w[i]), \forall\ i = 1\ to\ c$. As explained in Section 2.3.1, the problem with batching occurs here as well while performing the sum of the elements after the multiplication. The problem can be solved by the process of cyclically rotated and adding the encrypted vectors [20]. The steps (4–6) of Algorithm 6 describes the process of cyclically rotating and adding the r_i. The element in the first slot is the desired dot product. This operation is explained with an example in Fig. 4.

Homomorphic Comparison Protocol The procedure to compare two encrypted values without decryption is given in Algorithm 5. The gth bit, b_g of b is the comparison result, where $g = log_2 a + 1$ returned to the cloud server. If $b_g = 0$ then $m_1 < m_2$ otherwise $m_1 \geq m_2$.

The cmpsn protocol is secure because the protocol returns only one bit to the cloud server. Therefore, even in an attack scenario, the cloud server can only learn at most one single bit of the secret key. On the other hand, each time cmpsn protocol is invoked by PMCP:training protocol, the authentication server uses a new secret key. So, there will not exist any leakage of secret keys to the cloud server.

The steps (3–7) of Algorithm 6 repeats for c times yields c dot products which are assigned to $\varepsilon(R)$. The advantage of our method is that the server is unable to learn any information about the user's private data, i.e., reference templates or probe template as they are encrypted and the client device is not able to learn the information about the model parameters, i.e., weight vectors. Hence the privacy of both client device and model are preserved. The cloud server computes the dot products on the encrypted data; as a result, the privacy of the iris templates is achieved. If the cloud server did not perform the dot product result honestly and return a random result to minimize the use of its computational resources, then imposter may get access into the system. To overcome this limitation, the public verifier checks the correctness of the result returned by the cloud server.

2.3.4. Verification scheme for multi-class perceptron

The cloud server computes the dot products $\varepsilon(R) = r_i$, $\forall\ i = 1\ to\ c$ between encrypted weight vectors $\varepsilon(w[i])\ \forall\ i = 1\ to\ c$ and encrypted probe template $\varepsilon(Y)$. The verification scheme allows the public verifier to check the correctness of $\varepsilon(R)$ returned by the cloud server.

Generation of encrypted verification vector: After the enrollment phase, the authentication server constructs the encrypted verification vector using the weight vectors $\varepsilon(w[i]), \forall\ i = 1\ to\ c$ returned by the cloud server. The encrypted verification vector helps the public verifier to check the correctness of the dot product results.

Let $\varepsilon(Z_{n+1})$ be the encrypted verification vector and is defined as

$$\begin{aligned}\varepsilon(Z_{n+1}) &= \varepsilon(w[1]).\varepsilon(v_1) + \varepsilon(w[2]).\varepsilon(v_2) + \cdots . + \varepsilon(w[c]).\varepsilon(v_c) \\ &= \sum_{i=1}^{c} (\varepsilon(w[i]).\varepsilon(v_i))\end{aligned} \tag{6}$$

Algorithm 4 Perceptron for multi-class classification on encrypted data (PMCP:training)

Input: $\varepsilon(X_1)$, $\varepsilon(X_2)$, . . . , $\varepsilon(X_N)$, Corresponding class labels cls_1, cls_2, . . . ,cls_N, Iteration number T (Not Encrypted), BIAS=1 (Not Encrypted)

Output: The encrypted weight vectors for each class, $\varepsilon(w[i])$ where i ranges from 1 to c

begin
  for $i \leftarrow 1$ `to` c **do**
    $classes[i] \leftarrow i$
  end for
  for $i \leftarrow 1$ `to` c **do**
    for $j \leftarrow 1$ `to` d+1 **do**
      $w_{i,j} \leftarrow 1$
    end for
  end for
  for $i \leftarrow 1$ `to` c **do**
    $\varepsilon(w[i]) \leftarrow Enc(w_i, \delta_e)$ *//Batch Encryption of weight vectors*
  end for
  for T *iterations* **do**
    for $j \leftarrow 1$ `to` N **do**
      $arg\_max \leftarrow 0$
      $p\_cls \leftarrow classes[0]$
      $\varepsilon(arg\_max) \leftarrow Enc(arg\_max, \delta_e)$
      for $i \leftarrow 1$ `to` c **do**
        $ct \leftarrow multiply(\varepsilon(x_j), \varepsilon(w[i]))$
        for $k \leftarrow 0$ `to` p **do** *// where* $p = log_w^q$
          $ct \leftarrow ct + k_{g^k}(ct)$
        end for *//The element in the first slot is the desired dot product result*
        $b_z \leftarrow cmpsn(ct, \varepsilon(arg\_max))$
        if $b_z = 1$ **then**
          $\varepsilon(arg\_max) \leftarrow ct$
          $p\_cls \leftarrow i$
        end if
      end for
      if $cls_j \neq p\_cls$ **then**
        $\varepsilon(w[cls_j]) \leftarrow add(\varepsilon(w[cls_j]), \varepsilon(x_j))$
        $\varepsilon(w[p\_cls_j]) \leftarrow sub(\varepsilon(w[p\_cls_j]), \varepsilon(x_j))$
      end if
    end for
  end for
end

Algorithm 5 Homomorphic Comparison (cmpsn)

Procedure $cmpsn(C_1, C_2)$

Input: Ciphertexts C_1, C_2

Output: b_g

begin
  Compute $C_b = C_a + C_1 - C_2$ *//C_a is the encrypted value of a*
  $b = Dec(\delta_d, C_b)$
return b_g *//b_g is the g^{th} bit of b, where $g = \log_2 a + 1$*
end

where, $v_i \ \forall \ i = 1$ *to* c are the random integers and $\varepsilon(V) = (\varepsilon(v_1), \varepsilon(v_2), \ldots, \varepsilon(v_c))$. As long as the secret key is secure, encrypted verification vector is also secure and its security relies on the hardness of RLWE.

Ensuring the correctness of dot product: The public verifier checks the correctness of dot products $\varepsilon(R)$ using $\varepsilon(Z_{n+1})$, $\varepsilon(Y)$ and $\varepsilon(V)$. Our verification scheme checks the correctness of the result on the encrypted data itself as a result anyone can perform the correctness of the $\varepsilon(R)$ without the private information. The public verifier computes $\varepsilon(D1) = \varepsilon(Z_{n+1}).\varepsilon(Y)$ and $\varepsilon(D2) = \sum_{i=1}^{c}(r_i.\varepsilon(v_i))$. Finally, compute $(\varepsilon(D1) - \varepsilon(D2))$. If the result is zero, then the dot product values $\varepsilon(R)$ returned by the cloud server is considered to be correct. Eq. (7) uses Eq. (5), Eq. (6) and some algebraic properties of vectors and explains how $\varepsilon(D1)$ and $\varepsilon(D2)$ are same.

Algorithm 6 Perceptron for Multi-class Classification on encrypted data (PMCP:classification)

Input: $\varepsilon(Y)$, $\varepsilon(w[i])$ from training phase where i ranges from 1 to c
Output: $\varepsilon(R)$

begin
  for $i \leftarrow 1$ to c **do**
    $r_i \leftarrow multiply(\varepsilon(Y), \varepsilon(w[i]))$
    for $j \leftarrow 0$ to p **do** // *where* $p = log_w^q$
      $r_i \leftarrow r_i + k_{g^j}(r_i)$
    end for //*The element in the first slot is the desired dot product result*
    $r_i \leftarrow r_i$
  end for
  $\varepsilon(R) = (r_1, r_2, ..., r_c)$
return $\varepsilon(R)$
end

If the verification succeeds then the dot products $\varepsilon(R)$ are correct. So, the authentication server computes the predicted class by computing the index of the maximum value among $\varepsilon(R)$. The computed predicted class is compared with id given by the end user to determine whether the user is genuine or not.

$$\begin{aligned}\varepsilon(D1) &= \varepsilon(Y).\varepsilon(Z_{n+1}) \\ &= \varepsilon(Y).\sum_{i=1}^{c}(\varepsilon(w[i]).\varepsilon(v_i)) \\ &= \sum_{i=1}^{c}(\varepsilon(w[i]).\varepsilon(Y).\varepsilon(v_i)) \\ &= \sum_{i=1}^{c}\varepsilon(v_i).(\varepsilon(w[i]).\varepsilon(Y)) \\ &= \sum_{i=1}^{c}\varepsilon(v_i).r_i \\ &= \varepsilon(D2)\end{aligned} \tag{7}$$

3. Implementation details and security analysis

The following measures are used to evaluate the efficiency of a biometric system according to biometric information protection.[1]

1. Performance evaluation in terms of Equal Error Rate (EER), d-prime and KS-test.
2. Irreversibility and Unlinkability Analysis.
3. Computational cost.

3.1. Performance evaluation

Test Environment: We have compiled PNN and PMCP using Python 3.5.2 on Ubuntu 14.04 system. To validate the methods, we ran the experiments on a system with a 2.40 GHz Intel i7 processor & 16 GB RAM.

Experimental Setup: The University of Salzburg tool kit [18] is used in SvaS to extract the iris template from the human eye. SvaS considers the right & left eye images of the same person as different subjects due to the dissimilarity of patterns between them. CASIA-V 1.0 consists of 108 subjects with 7 samples each. The CASIA-V3-Interval consists of 165 subjects of the right eye & 172 subjects of the left eye, each having five samples. The first five samples out of 10 samples in IITD & SDUMLA-HMT databases are considered. We performed multiple experiments with 40%–60% (S1), 60%–40% (S2) and 80%–20% (S3) training–testing partitions respectively.

Classification Accuracy: The classification accuracy of SvaS with PNN & PMCP for CASIA-V 1.0, CASIA-V3-Interval, IITD and SDUMLA-HMT with S1, S2 & S3 are shown in Fig. 5. The comparison of accuracy between protected & unprotected templates of SvaS for different databases when train–test split ratio is 60–40 is shown in Fig. 6. From Fig. 6, we infer that there is no degradation of accuracy between unprotected & protected templates in SvaS. Measures like EER, d-prime and KS-test are also considered to validate the efficiency of SvaS. KS-test and d-prime values are calculated to identify the separability of genuine and imposter scores. Fig. 7,

[1] https://www.iso.org/standard/52946.html.

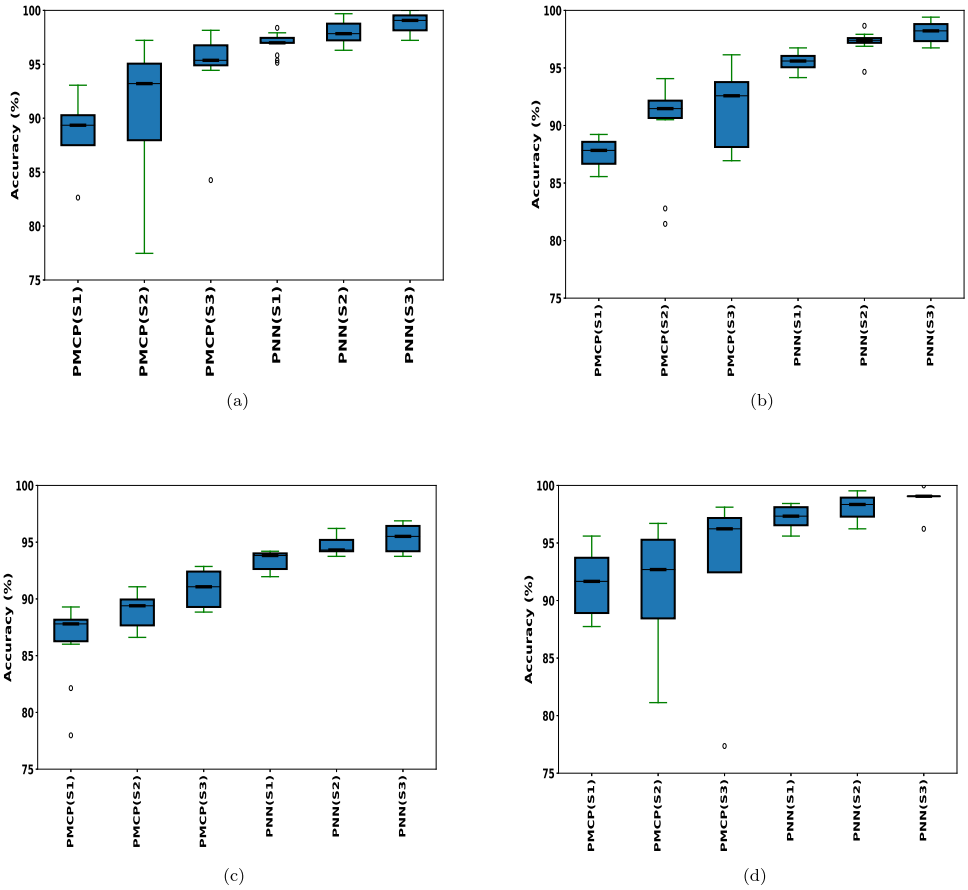


Fig. 5. Accuracy of SvaS (PMCP & PNN with different train–test split ratio) obtained for (a) CASIA-V 1.0 (b) CASIA-V3-Interval (c) IITD (d) SDUMLA-HMT iris databases.

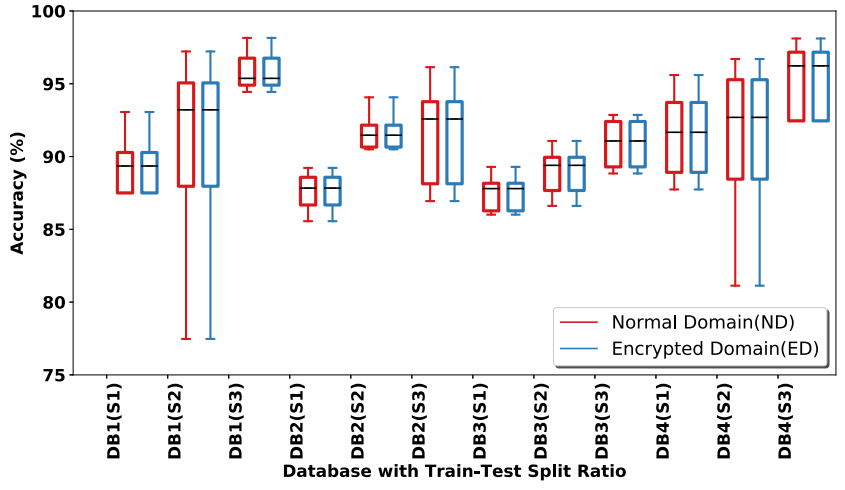
Fig. 8 shows the EER, d-prime, KS-test obtained for different databases of iris template with size 2560. The imposter and genuine scores are well separated if the d-prime value is larger. The range of KS-test value lies between 0 and 1. The imposter and genuine scores are well separated if KS-test value is closer to 1.

The statistical significant tests such as McNemar's test and paired T-test are considered to show the significance of MCP and NN with other classifiers such as Support Vector Machines (SVM), Decision Tree (DT), Random Forest (RF), and Naive Bayes (NB). The McNemar's and paired T-test values for different databases are given in Table 2. From Table 2, we can infer that the *p*-values are well below the accepted significance level of 0.05 except for SVM, which indicates that the performance improvement of MCP and NN over DT, NB and RF are statistically significant.

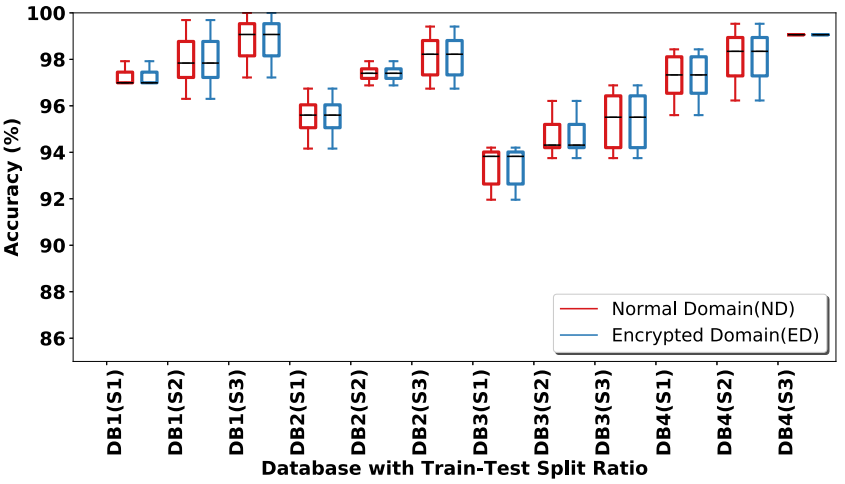
3.2. Security analysis

The template protection method must satisfy the requirements of irreversibility, revocability and unlinkability to ensure the privacy of the iris templates. The vulnerability of attacks in SvaS may happen in the entries namely, *the cloud server*, *the client device*, *the communication channel between the cloud server and the client device*, *the authentication server*, *the public verifier*. The features of the iris image is extracted by the client device. So, security is to be ensured for the client device. Since, SvaS assume the client device is a trusted entity, the features of iris image are secure. The authentication server generates the secret and public keys. SvaS assume that the authentication server is also a trusted entity. Since the security of SvaS depends on the apparent hardness of RLWE problem, the iris templates stored in the server database are secure. It is difficult to decrypt the encrypted iris templates without the secret key. As a result, the communication channel is also reliable.

Irreversibility Analysis: Irreversibility refers to obtaining the original template from the encrypted template. The client device sends the encrypted reference templates to the cloud server during the enrollment phase, and encrypted probe iris template of a user to the server for classification result. The server classifies the encrypted probe template and returns the encrypted classification result to the authentication server. As the SvaS uses BFV scheme to protect the templates, and the security of BFV scheme relies on



(a)



(b)

Fig. 6. Comparison of accuracy between protected and unprotected templates for a) MPC and b) NN; **DB1**: CASIA-V 1.0, **DB2**: CASIA-V3-Interval, **DB3**: IITD and **DB4**: SDUMLA-HMT.

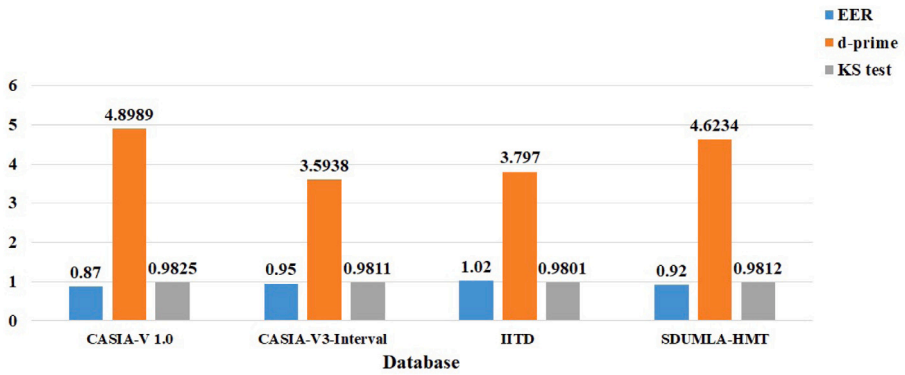


Fig. 7. EER, d-prime and KS-test for benchmark iris databases using NN.

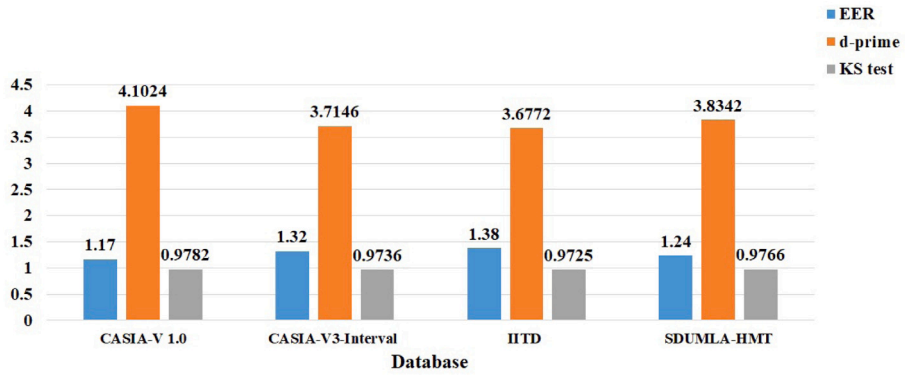


Fig. 8. EER, d-prime and KS-test for benchmark iris databases using MCP.

Table 2
Statistical significance results for SvaS.

	Model-1 vs. Model-2	CASIA-V 1.0		CASIA-V3-Interval		IITD		SDUMLA-HMT	
		T-value	*p*-value	T-value	*p*-value	T-value	*p*-value	T-value	*p*-value
Paired T-test	MCP vs. DT	14.32	.0009	11.32	.009	10.95	.004	12.84	.001
	MCP vs. SVM	.95	.3782	.68	.824	.64	.956	.75	.529
	MCP vs. NB	5.72	.00359	3.89	.00986	3.12	.00994	4.24	.00924
	MCP vs. RF	75.24	9.3e−05	72.58	8.2e−04	71.38	9.8e−04	73.82	3.2e−04
	NN vs. DT	40.24	3.4e−08	31.36	.00001	15.24	.00001	35.82	1.2e−06
	NN vs. SVM	3.25	.00924	.95	.3782	1.42	.12594	1.25	.16324
	NN vs. NB	7.24	.000527	3.64	.00659	7.52	.00072	5.72	.003529
	NN vs. RF	37.89	6.9e−07	30.54	.00001	79.24	2.3e−12	34.24	3.2e−06
		χ^2-value	*p*-value	χ^2-value	*p*-value	χ^2-value	*p*-value	χ^2-value	*p*-value
McNemar's test	MCP vs. DT	89.24	3.24e−21	86.24	1.2e−20	85.98	5.4e−20	87.24	8.23e−21
	MCP vs. SVM	1.4	.2892	.8	.4231	.73	.548	.9	.3924
	MCP vs. NB	25.32	5.39e−06	23.52	9.24e−06	22.89	1.89e−05	24.89	7.82e−06
	MCP vs. RF	108.59	8.34e−29	106.32	8.24e−28	105.18	1.48e−27	107.24	3.9e−28
	NN vs. DT	87.48	1.89e−20	80.01	3.72e−19	104.01	9.34e−25	84.23	2.52e−19
	NN vs. SVM	2.9	.0924	.5	.4795	.3	.6248	1.4	.2892
	NN vs. NB	14.24	.00048	8.47	.00361	29.45	7.38e−08	11.32	.000952
	NN vs. RF	84.24	2.35e−19	75.11	4.45e−18	140.01	3.54e−32	80.01	3.72e−19

Table 3
Security and timing parameters for PMCP and PNN.

	Security (λ)	d	No FHE time (s)	Parameters			Time with Batching scheme (s)				Time without Batching scheme (s)			
				n	q (bits)	a	Enc	Score	Dec	Total	Enc	Score	Dec	Total
PMCP	128-bit	640	0.16	1024	29	40 961	0.003	1.19	0.0008	1.1938	2.688	482.2615	0.64	485.59
		1280	0.3	2048	56	40 961	0.005	2.1	0.0018	2.1068	5.376	964.5250	1.28	971.181
		2560	0.38	4096	110	40 961	0.011	4.13	0.0038	4.1448	12.544	1929.0498	4.096	1945.690
	192-bit	640	0.16	1024	20	40 961	0.004	1.21	0.0009	1.2149	2.688	483.2615	0.768	486.7175
		1280	0.3	2048	39	40 961	0.005	2.2	0.0016	2.2066	5.504	964.825	1.664	971.993
		2560	0.38	4096	77	40 961	0.013	4.15	0.0040	4.167	12.8	1930.212	4.096	1947.108
PNN	128-bit	640	0.15	1024	29	40 961	0.003	2.24	0.0008	2.2438	2.688	680.393	0.64	683.721
		1280	0.27	2048	56	40 961	0.005	4.4	0.0018	4.4068	5.376	1360.786	1.28	1367.442
		2560	0.52	4096	110	40 961	0.011	13.3	0.0038	13.3148	12.544	2721.603	4.096	2738.243
	192-bit	640	0.15	1024	20	40 961	0.004	2.12	0.0009	2.1249	2.688	681.3421	0.768	684.849
		1280	0.27	2048	39	40 961	0.005	4.4	0.0016	3.4666	5.504	1360.796	1.664	1367.964
		2560	0.52	4096	77	40 961	0.013	13.71	0.0040	13.727	12.8	2722.603	4.096	2739.499

Table 4
Baseline comparison (Iris template is of 1 × 2560 dimension).

Database	Template type	Accuracy (%)	EER (%)	Time
CASIA-V 1.0	Classification using MCP on unprotected iris templates	94.44	1.17	0.38
	Classification using PMCP on protected iris templates with λ = 128-bit	94.44	1.17	4.1448
	Classification using PMCP on protected iris templates with λ = 192-bit	94.44	1.17	4.167
	Classification using NN on unprotected iris templates	98.81	0.87	0.52
	Classification using PNN on protected iris templates with λ = 128-bit	98.81	0.87	13.3148
	Classification using PNN on protected iris templates with λ = 192-bit	98.81	0.87	13.727
CASIA-V3-Interval	Classification using MCP on unprotected iris templates	91.52	1.32	0.38
	Classification using PMCP on protected iris templates with λ = 128-bit	91.52	1.32	4.1448
	Classification using PMCP on protected iris templates with λ = 192-bit	91.52	1.32	4.167
	Classification using NN on unprotected iris templates	98.12	0.95	0.52
	Classification using PNN on protected iris templates with λ = 128-bit	98.12	0.95	13.3148
	Classification using PNN on protected iris templates with λ = 192-bit	98.12	0.95	13.727
IITD	Classification using MCP on unprotected iris templates	90.89	1.38	0.38
	Classification using PMCP on protected iris templates with λ = 128-bit	90.89	1.38	4.1448
	Classification using PMCP on protected iris templates with λ = 192-bit	90.89	1.38	4.167
	Classification using NN on unprotected iris templates	97.35	1.02	0.52
	Classification using PNN on protected iris templates with λ = 128-bit	97.35	1.02	13.3148
	Classification using PNN on protected iris templates with λ = 192-bit	97.35	1.02	13.727
SDUMLA-HMT	Classification using MCP on unprotected iris templates	92.26	1.24	0.38
	Classification using PMCP on protected iris templates with λ = 128-bit	92.26	1.24	4.1448
	Classification using PMCP on protected iris templates with λ = 192-bit	92.26	1.24	4.167
	Classification using NN on unprotected iris templates	98.68	0.92	0.52
	Classification using PNN on protected iris templates with λ = 128-bit	98.68	0.92	13.3148
	Classification using PNN on protected iris templates with λ = 192-bit	98.68	0.92	13.727

solving the RLWE problem, it is difficult to decrypt the templates by the server or an imposter without secret key (δ_d). Therefore, SvaS satisfies the irreversibility property.

Revocability Analysis: Revocability ensures that a new protected template should be generated by the protection method if the old template is compromised or stolen. In SvaS, Revocability can be achieved by re-encrypting the samples in the database with a new key pair (δ'_e, δ'_d) instead of acquiring the new samples from the users.

Unlinkability Analysis: Unlinkability ensures that there will not be any correlation between the protected templates used in different applications. BFV scheme used in SvaS is based on probabilistic encryption. Due to the randomness involved in BFV scheme, different ciphertexts can be generated even if the same message is encrypted multiple times with the same key, and there will not exist any similarity between the generated ciphertexts.

3.3. Computational analysis

The security parameters used in SvaS are polynomial modulus (x^n+1), coefficient modulus (q), plaintext modulus (a) and security level (λ). We considered two different values for λ. From Table 3, it can be inferred that the higher security level has nearly no influence on the execution time. $x^n + 1$ must be a power-of -2 cyclomatic polynomial, i.e., of form $x^2 + 1$. The security level is directly proportional to the polynomial modulus. On the other hand, larger $x^n + 1$ makes ciphertext size larger, and all operations become slower. n value must be a power of 2 and greater than the size of the iris template. So, we choose different n values for different sizes of iris templates. a can be any positive integer, and mostly it is a power of two. But, batching encoding only works when a is chosen to be a prime number and congruent to 1 (mod $2n$). So, we considered plaintext modulus as 40 961.

Table 3 shows the time taken (in seconds) to encrypt, decrypt and to classify $\varepsilon(Y)$ for various sizes of iris templates with different security parameters. The experiments are run by ten times and the average time is considered. The table also shows the time taken to perform classification on unencrypted values. From Table 3, we infer that the reduction in the size of the iris template and batching scheme can speed-up homomorphic iris computation over element-wise (without batching scheme). The iris template size is proportional to the computational time. SvaS converts 1 × 10 240 into 1 × 640, 1 × 1280, 1 × 2560 respectively. Even though the total time taken for iris code of size 640, and 1280 is less when compared to iris code of size 2560, but the optimal accuracy is achieved with iris template of size 1 × 2560. The higher security level has nearly no influence on the execution time.

3.4. Comparison analysis

Table 4 shows the baseline comparison of overall average classification accuracy and average computational time between unprotected and protected templates. During enrollment, the time or space is not restricted. Therefore, we did not consider the time taken to encrypt the reference templates, $\varepsilon(X_i)$ and time taken to perform training of classifiers on encrypted templates. For the protected templates, we consider the overall average time, i.e., a sum of encryption time, classification time, verification time

Table 5
Comparison of SvaS with the existing approaches (in terms of accuracy).

Method	CASIA-V3-Interval	IITD
Sardar et al., [21]	97.12%	97.19%
Barpanda et al., [22]	91.65%	89.72%
Arsalan et al., [14]	99.10%	98.41%
Zhao et al., [15]	96.92%	96.80%
Noruzi et al., [23]	98.80%	99.57%
SvaS	98.12%	97.35%

Table 6
Comparison of SvaS with existing approaches (in terms of separability measure (d-prime)).

	CASIA-V 1.0	CASIA-V3-Interval	IITD
Sadhya, D. et al. [24]	–	2.39	2.92
Barpanda. et al. [22]	–	1.71	1.76
Walia, G.S. et al. [25]	2.6053	–	1.9578
SvaS (NN)	4.8989	3.5938	3.797
SvaS (MCP)	4.1024	3.7146	3.677

and decryption time. From Table 4, we infer that there is no accuracy loss between protected and unprotected templates for NN and MCP. Even though the computational time of authentication on unencrypted templates takes less time when compared to the authentication on encrypted templates, the advantage of SvaS is that it satisfies the properties of the biometric template protection scheme. Hence, SvaS is secure. The increase in computational time is due to the enhanced functionalities of the machine learning algorithms (PMCP and PNN) used in SvaS.

Table 5 shows the accuracy comparison of SvaS with the other existing works for CASIA-V3-Interval and IITD databases. SvaS shows a better accuracy when compared to [15,21,22] and lesser accuracy when compared to [14,23], which are devoid of guarantee the requirements of biometric template protection schemes. The d-prime comparison of SvaS with the existing approaches are shown in Table 6. We can infer from Table 6 that the imposter and genuine scores are well separated. The classification on the encrypted templates guarantees the requirements of biometric template protection schemes without a drop in the accuracy. SvaS guarantees the irreversibility, diversity & performance degradation and additionally, SvaS verifies the result computed by the cloud server to avoid false accept/reject.

4. Conclusion

A secure and verifiable machine learning-based iris authentication system using fully homomorphic encryption on a malicious cloud server is proposed. Our method is the first known iris authentication system where training and testing are performed on the encrypted templates by using private multi-class perceptron and private nearest neighbor algorithms. Our method mitigates the problem of trusting a third party or cloud server for classification. The idea is to generate an encrypted verification vector which checks the correctness of the result returned by the cloud server. We experimented our method on four benchmark publicly available iris databases to check the efficiency. The experimental results show that the accuracies of the nearest neighbor and multi-class perceptron on protected and unprotected templates are the same, which proves the effectiveness of the system. The training and classification are performed on encrypted templates; as a result, the training and test templates are known only to the client device, and the model is accessible only to the server. Our method fulfills all the requirements of biometric information protection.

Private nearest neighbor and private multi-class perceptron can experiment on other biometric traits like a fingerprint, face, finger-vein, etc. to ensure the privacy of both the user's data and the model. Further research has to be done in terms of the public verifier directly checks the classification result. Since the template size is directly proportional to the computation time, an optimal algorithm has to be proposed in the future work which compresses the templates without compromising the accuracy.

CRediT authorship contribution statement

Mahesh Kumar Morampudi: Conceptualization, Methodology, Software, Data curation, Writing - original draft, Writing - review & editing. **Munaga V.N.K. Prasad:** Methodology, Writing - review & editing, Supervision. **Mridula Verma:** Validation, Investigation, Supervision. **U.S.N. Raju:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Pirbhulal S, Shang P, Wu W, Sangaiah AK, Samuel OW, Li G. Fuzzy vault-based biometric security method for tele-health monitoring systems. Comput Electr Eng 2018;71:546–57.
[2] Daugman J. How iris recognition works. IEEE Trans Circuits Syst Video Technol 2004;14(1):21–30. http://dx.doi.org/10.1109/TCSVT.2003.818350.
[3] Galbally J, Ross A, Gomez-Barrero M, Fierrez J, Ortega-Garcia J. Iris image reconstruction from binary templates: An efficient probabilistic approach based on genetic algorithms. Comput Vis Image Underst 2013;117(10):1512–25.
[4] Fontaine C, Galand F. A survey of homomorphic encryption for nonspecialists. EURASIP J Inf Secur 2007;2007(1):15–25.
[5] Šeděnka J, Govindarajan S, Gasti P, Balagani KS. Secure outsourced biometric authentication with performance evaluation on smartphones. IEEE Trans Inf Forensics Secur 2014;10(2):384–96.
[6] Haghighat M, Zonouz S, Abdel-Mottaleb M. CloudID: Trustworthy cloud-based and cross-enterprise biometric identification. Expert Syst Appl 2015;42(21):7905–16.
[7] Xiang C, Tang C, Cai Y, Xu Q. Privacy-preserving face recognition with outsourced computation. Soft Comput 2016;20(9):3735–44.
[8] Gomez-Barrero M, Maiorana E, Galbally J, Campisi P, Fierrez J. Multi-biometric template protection based on homomorphic encryption. Pattern Recognit 2017;67:149–63.
[9] Zhou K, Ren J. Passbio: Privacy-preserving user-centric biometric authentication. IEEE Trans Inf Forensics Secur 2018;13(12):3050–63.
[10] Barni M, Droandi G, Lazzeretti R, Pignata T. SEMBA: secure multi-biometric authentication. IET Biom 2019;8(6):411–21.
[11] Rai H, Yadav A. Iris recognition using combined support vector machine and Hamming distance approach. Expert Syst Appl 2014;41(2):588–93.
[12] Ahmadi N, Akbarizadeh G. Hybrid robust iris recognition approach using iris image pre-processing, two-dimensional gabor features and multi-layer perceptron neural network/PSO. IET Biom 2017;7(2):153–62.
[13] Ahmadi N, Nilashi M, Samad S, Rashid TA, Ahmadi H. An intelligent method for iris recognition using supervised machine learning techniques. Opt Laser Technol 2019;120:1–11.
[14] Arsalan M, Kim DS, Lee MB, Owais M, Park KR. FRED-Net: Fully residual encoder–decoder network for accurate iris segmentation. Expert Syst Appl 2019;122:217–41.
[15] Zhao Z, Kumar A. A deep learning based unified framework to detect, segment and recognize irises using spatially corresponding features. Pattern Recognit 2019;93:546–57.
[16] Sun X, Zhang P, Liu JK, Yu J, Xie W. Private machine learning classification based on fully homomorphic encryption. IEEE Trans Emerg Top Comput 2020;8(2):352–64.
[17] Graepel T, Lauter K, Naehrig M. ML confidential: Machine learning on encrypted data. In: International conference on information security and cryptology. Springer; 2012, p. 1–21.
[18] Rathgeb C, Uhl A, Wild P, Hofbauer H. Design decisions for an iris recognition sdk. In: Handbook of iris recognition. Springer; 2016, p. 359–96.
[19] Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption. IACR Cryptol ePrint Arch 2012;2012:1–19.
[20] Gentry C, Halevi S, Smart NP. Fully homomorphic encryption with polylog overhead. In: Annual international conference on the theory and applications of cryptographic techniques. Springer; 2012, p. 465–82.
[21] Sardar M, Mitra S, Shankar BU. Iris localization using rough entropy and CSA: A soft computing approach. Appl Soft Comput 2018;67:61–9.
[22] Barpanda SS, Sa PK, Marques O, Majhi B, Bakshi S. Iris recognition with tunable filter bank based feature. Multimedia Tools Appl 2018;77(6):7637–74.
[23] Noruzi A, Mahlouji M, Shahidinejad A. Iris recognition in unconstrained environment on graphic processing units with CUDA. Artif Intell Rev 2019;1–25.
[24] Sadhya D, Raman B. Generation of cancelable iris templates via randomized bit sampling. IEEE Trans Inf Forensics Secur 2019;14(11):2972–86.
[25] Walia GS, Rishi S, Asthana R, Kumar A, Gupta A. Secure multimodal biometric system based on diffused graphs and optimal score fusion. IET Biom 2019;8(4):231–42.

Morampudi Mahesh Kumar is currently a doctoral candidate in the Department of Computer Science and Engineering at National Institute of Technology, Warangal, India and Institute for Development & Research in Banking Technology, Hyderabad, India. His research interests include Biometrics, Cryptography and Privacy in Machine Learning. He is a life member of ISTE, CSI.

Munaga V.N.K. Prasad received his doctoral degree from the Institute of Technology, Banaras Hindu University, India. Currently, he is an Associate Professor at the Institute for Development and Research in Banking Technology, Hyderabad, India. His research interests include Biometrics, Payment System Technologies and Data Hiding. He is a senior member of IEEE and ACM.

Mridula Verma received his doctoral degree from the Institute of Technology, Banaras Hindu University, India. Currently, she is working as Assistant Professor in Institute for Development and Research in Banking Technology, Hyderabad, India, Hyderabad. Her research interests include Machine Learning optimization algorithms and applications, Convex Optimization, and Data Science.

U.S.N. Raju received his doctoral degree from JNT University Kakinada, India. He worked in industry for two years and then in academics for 15+ years. Presently he is working in department of Computer Science and Engineering at National Institute of Technology Warangal, Telangana, India. He is a Life member of IEEE, ISTE, CSI, ISCA and IEI.