iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** III **Month of publication:** March 2022

DOI: https://doi.org/10.22214/ijraset.2022.40570

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue III Mar 2022- Available at www.ijraset.com

Privacy Preserving Bio-Metric Authentication

Karan Ladhiya
Institute Of Technology, Nirma University, Ahmedabad, India

Abstract: Biometric authentication is now extensively used in several systems and applications to authenticate users using their biometrics. The biometrics of the users are compared to the biometric templates already stored on the server, and if there is a match, only the user is permitted to enter the system. However, because each user's biometrics are unique, it is more important than the user's actual biometric data is never leaked. Moreover, the utilization of the user's actual biometric data for comparisons during the authentication process can't be done because the revelation of the user's actual biometrics to the server should not be done. Throughout authentication, each user will encrypt his biometrics and then transmit this encrypted data to the server for comparison, and this data will never be decrypted for privacy reasons during the whole authentication process. To compare two encrypted data without decrypting them, the present study uses the homomorphic properties of the Pailler cryptosystem which will be the encryption of the algorithm for the comparison part. The use of Euclidean Distance is made to find the squared distance between the users' queried feature vector and the templates stored into the server. In the end, among all the distances, the minimum distance will be chosen and will compare with some predefined threshold to decide whether the user is an authenticated user or not.

I. INTRODUCTION

Biometrics authentication has gained very much importance in recent years, this authentication process uses face, fingerprints, eyes, etc as biometric data. The user has to first register into the system and while registering, he will give his biometric templates which will be stored on the server. The user mostly gives more than one copy of its biometric, for example, in the case of face recognition, the user will be asked to give different images having different expressions and taken from different angles so that the authentication is done in fewer trials as possible [1-3].

During the authentication, the user's actual biometrics must be kept secret from the server so that if during the authentication the data gets leaked, then it would be impossible for that malicious party to get the original data of the user in polynomial time. For that, the encryption schemes will be used which we will deal in the numbers that will be of several of hundreds of digits. Also, decryption of the encrypted data of the user at any point of time during the authentication can't be done. To pursue that, the use of Paillier's encryption scheme is made because of its additive homomorphic property. Because of this property, the present study will be able to compare two encrypted data without decrypting them. It will have to maintain the privacy of the templates stored in the server from the user. Also, the permission of letting the server know which one of its biometric templates is matched with the user's queried feature vector because of the same possibility of the server getting breached during the authentication must be prohibited [4-6].

The use of Euclidean distance is made in order to find the closeness between the user's data and the templates stored on the server. Yet again all the operations that will be done to compute the distance will be in an encrypted domain i.e., not be decrypting the user's data to compute the distance. Then, a comparison among these encrypted distances to find the minimum of them is to be done. Afterward, this minimum encrypted distance will be compared with some threshold value which can be decided by doing several experiments. If this minimum encrypted distance is less than or equal to the threshold, then it will let the user enter into the system for being an authenticated user. Before all these, the system has to extract the features from the images that are stored during the enrollment process of the user. Before storing an image into the server also, the feature extraction is performed and then that feature vector will be stored in the server.

A. Problem Statement

At the server, we have got N templates $X = \{x_1, x_2, \ldots, x_N\}$ stored and each template is a vector of dimension d that is, each template is having d features. During the registration or enrollment of the user, we converted his actual image into a feature vector by performing all the feature extractions, and these final feature vectors are represented as $\boldsymbol{x}_i$ here for $1 \leq i \leq N$. For the client, we have got the user's actual image $W_{[1xD]}$ of dimension D which will be encrypted according to the Paillier encryption scheme before sending to the server. After encryption of the actual image of the user, we will send this to a server for an authentication process. At the server, the Encrypted image will be transformed into an encrypted feature vector which is represented as $[\boldsymbol{u}_{[1xd]}]$. Any encrypted entity will be represented as [.]. We will use this encrypted feature vector for all calculations [7-9].

The server will have to tell after doing the comparison of this the user's feature vector with a template of servers, whether the user has authenticated the user or not. The server will return one boolean value only, telling whether the user is valid or not. Other than that, no information regarding the user will be returned.

B. Approach

The problem of PPBA mainly consists of four modules

1) *Encryption:* In this module, we will have the user's feature vector that will be encrypted using Paillier encryption and then this encrypted feature vector will be sent to the server. A server, for performing the computations in the next module, we will be needed to do encryption on the server-side as well using the same encryption scheme. These encrypted feature vectors will be used as input for the next module [10].
2) *Feature Extraction:* In this module we will have the actual image of the user of dimension D. We will perform some dimension reduction and will extract some features from the image transforming the D dimensional vector into the feature vector of dimension d where d << D. Output of this module will be the final feature vector which will be used as an input for the next module. Also, all the templates that will be stored on the server, will pass through this module first [11].
3) *Matching:* This module is consist of two things. First, the server will compute the distance between the user's feature vector and templates of the server in an encrypted domain. Then the second thing is to compare this distance and to choose the minimum among them. This minimum distance will be used as the input for the next module [9].
4) *Decision:* This module is the last one and is rather simple. We will just compare the minimum distance with the predefined threshold. And based on this comparison, the server will decide whether the user is valid or not [8].

II. ENCRYPTION MODULE

For our research, we have used the Paillier Encryption scheme. It is a Public-key Cryptosystem based encryption scheme and it was introduced by Paillier. In the Paillier encryption scheme, we first randomly generate two huge prime numbers p and q. These numbers will be of several hundreds of decimal digits. Then we set n = pq. We also set λ = lcm (p-1, q-1) where lcm is the least common multiple.

Then according to the encryption scheme, we have to choose a random number g such that L(g^{λ} mod n^2) is co-prime with n i.e. GCD (n, L(g^{λ} mod n^2))=1, where L(x) = $\frac{x-1}{n}$ and GCD is the greatest common divisor. Here the L(x) will take the floor value of the output of division. This condition will be true if we take p and q of the same length and take g = n + 1. So, to avoid checking this condition every time, we will take p and q of the same length and g = n + 1. Then again, we select a random number r such that r<n.

Now if I denote the plain text or original text as m, then the corresponding ciphertext c or encrypted value will be,

$$c = g^m r^n \ mod \ n^2 \quad (1)$$

Here x mod y gives the reminder when y divides x. As it is a public-key encryption scheme, there will be a public and private key. In Paillier, (n, g) combined is the public key whereas (p, q) or λ is the private key. The public key will be made public so that anyone can encrypt the plain text using this key but the only owner of the private key can decrypt it.

The decryption of given ciphertext for Paillier is done using the below-given formula,

$$m = \frac{L(c^{\lambda} mod\ n2)}{L(g^{\lambda} \ mod \ n^2)} mod \ n \quad (2)$$

As λ is the private key, only the owner of that key can decrypt any ciphertext.

The main reason behind choosing this particular encryption scheme is its additive homomorphic property. The homomorphic property allows us to do computations or operations on the ciphertext or encrypted data. The result of this computation or operation will also be a ciphertext but when we decrypt it, it gives the same result as if the operations had been performed on the plain text. In the case of Paillier encryption, it is homomorphic for the operation of addition. That is, if you multiply two ciphertexts and then decrypt the result of multiplication, then it will be the same as the addition of the actual plain text.

$$(m_1 + m_2) \ mod \ n = D(E(m_1) \ E(m_2) \ mod \ n^2) \quad (3)$$

Where m_1, m_2 are two plain texts and E(.) and D(.) denotes the encryption and decryption respectively.
This property is important because of the requirement that we can not decrypt the ciphertexts during the process of authentications for security concerns. The above-given expression is not restricted to two operands, we can use more than two operands as well. Because of that, here are some variants of the above-given expressions

$$(m_1 m_2)\ mod\ n = D(E(m_1)^{E(m_2)}\ mod\ n^2) \quad (4)$$

$$(m_1 m_2)\ mod\ n = D(E(m_2)^{E(m_1)}\ mod\ n^2) \quad (5)$$

$$(k m_2)\ mod\ n = D(E(m_1)^{k}\ mod\ n^2) \quad (6)$$

Where k is some scalar. In PPBA, we will mostly have a feature vector to encrypt which is a 1-D matrix of dimension d. To encrypt the entire vector, we will have to encrypt each element of it using the same public keys. In the case of a client, he will send this encrypted vector [u] only to the server for authentication. Also, the client will generate the public and private keys and will publish the public key. So, at any point of time during authentication, if the server needs something to decrypt (of course he will not be decrypting something that will expose the actual user biometrics during the authentication) then he will need the help of the client for that because no one other than the client has got the private key and hence no one other than the client can decrypt anything. For more details about the Paillier scheme, we would refer the reader to this (1) original paper.

III. FEATURE EXTRACTION MODULE

A. *PCA for Feature Extraction Process*

Feature extraction technique of Image is Dimensionality Reduction of Image. We have already discussed the database that we used in this research is the "ORL face database". In this database, the Dimensionality of each image is 92 x 112. In the next few sections, we will see that in this authentication process we have to do many complex operations in Paillier Cryptosystem. And if we work with images of 92 x 112 then it will be very costly in terms of execution time. So, we have to transform face images into a characteristic feature vector of low-dimensional vector space(Dimensionality Reduction). We will call low-dimensional vector space face space. This will eventually help us to reduce execution time for the authentication process. How? we will see later.

There are many algorithms for extracting features from an image like LDA(Linear Discriminant Analysis), ICA(Independent Component Analysis), PCA(Principal Component Analysis). We used PCA for the feature extraction process. Initially, we have to divide the "ORL FaceImage database" into two sets:

1) *Training Set:* When a user is enrolling herself/himself into the system, we stored images in a server-side database. These images are called Training Set. Also, We used this set in PCA.
2) *Test Set:* When a user wants to enter into the system (During the authentication process), we capture an image of the user. And a set of captured images is called a test set.

FaceSpace is consists of some basis vectors(eigenfaces) that we will have to find. Now, we will see the procedure of finding eigenfaces using PCA.

We have looked at the algorithm of extracting features from training images using PCA. We used Python programming language to implement this algorithm.

a) *Input:* During enrollment process, We have stored N training images $\boldsymbol{T}_{[DxN]} = [\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3 \ldots \boldsymbol{t}_N]$. Where, $\boldsymbol{t}_i \in IR^D$. Here, we will project our images onto these eigenfaces and we will get image transformation from D-dimension to d-dimension. where, d << D. This way, we will do dimensionality reduction.
b) *Procedure:* Many FaceSpace are consisting of d Basis vectors. But we have to find a FaceSpace that will reduce dimensionality without losing much information. For that, Our objective is to find a FaceSpace(Y_{Dxd}) that has maximum variance when we project an image onto it.

$$\max_{all\_possible\_Y} Var(Y^T T) \quad (7)$$

After applying optimization strategies like Langrage, We concluded that eigenfaces are nothing but eigenvectors of *Cov*(*T*). But the important thing is "which eigenvector do we have to choose?". We will get at most D-eigenvectors that have non-zero eigenvalue. For any eigenvector $e_i \in IR^D$.

$$Var(e_i^T T) = \lambda_i \quad (8)$$

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue III Mar 2022- Available at www.ijraset.com

where, λ_i is eigenvalue of corresponding eigenvector. We can see the most important part is that not all eigenvector contribute equally to variance when we project an image onto the FaceSpace. So, our first eigenface will be that eigenvector that has the highest corresponding eigenvalue. Here, we calculate *Cov*(*T*).

$$\tilde{T} = T - \bar{m} \quad (9)$$

$$\overline{m_i} = \frac{1}{N}\sum_{j=1}^{N} T(i.j) \quad (10)$$

Where each $\bar{m}_i$ denotes the mean of pixel i of all training images. And $\bar{M}\ 1xD = [\bar{m}_i, \bar{m}_i\ \bar{m}_3 ... \bar{m}_D]$ is a vector of a mean of each pixel.

$$Cov(T)_{DxD} = \tilde{T}DxN\ .\ tranpose(\tilde{T})_{NxD} \quad (11)$$

Where, $U_{[DxD]}$ is matrix of Eigenvector. There are D-eigenvector of D-dimension. $S_{[DxD]}$ is diagonal matrix and *S* (*i*, *i*) will be eigenvalue of $U_{i[Dx1]}$. These eigenvalues in matrix *S* are stored in decreasing order.

$$S\ VD[Cov(T)] = US\ V \quad (12)$$

Where, $U_{[DxD]}$ is matrix of Eigenvector. There are D-eigenvector of D-dimension. $S_{[DxD]}$ is diagonal matrix and *S* (*i*, *i*) will be eigenvalue of $U_{i[Dx1]}$. These eigenvalues in matrix *S* are stored in decreasing order.

Finally, We got eigenvector aka eigenfaces. The span of d- eigenvector is called FaceSpace. From now onwards, we will use $Y_{[dxD]}$ (Matrix of eigenfaces) transformation matrix to transform any Images into a feature vector(Lower-dimension space).

In the next module, we have to do the encryption and decryption process. In a Cryptographic environment, we must use integer numbers. But in this module transformation matrix consist of a non-integer value. Also when we find an eigenvector using SVD, each eigenface has a unit norm implies that we cannot directly round that to the nearest integer. If we do that then most of the values will be 0 which is not acceptable. We have found one solution to this problem.

We will multiply each eigenface with a constant(Norm of eigenface) which shifts the values and then if we round off the non-integer value to the nearest integer will give us some meaningful numbers. We used a constant value of norm = 1000.

Here is the final important part of this module "How many feature vectors we should use such that along with dimensionality reduction error should be as low as possible". Parameter d is heavily dependent on the dataset(In this case, we have FaceImage of ORL database). We will choose the value of d based on some analysis like "How much information each eigenvector has?", "If we won't consider some eigenvectors how much error do we get as compared to the original image?". We plotted some graphs for analysis purposes and on basis of that, we chose the value of d = 12.

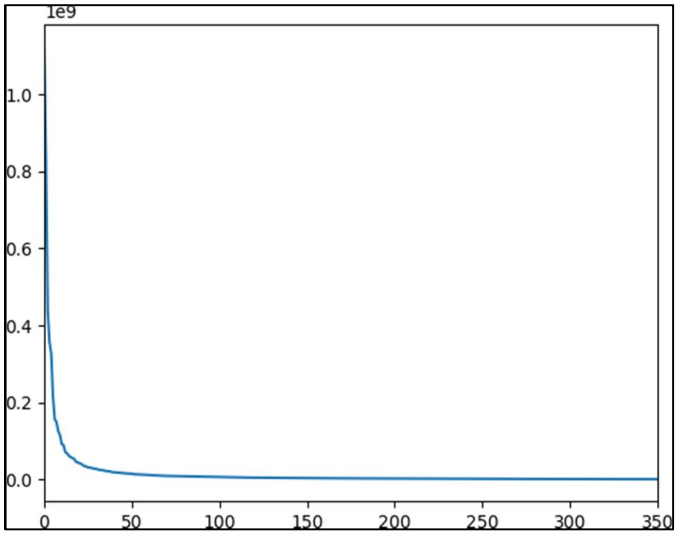


Figure 1: Projection of the encrypted query image into facescape that is generated using PCA

The first thing that server has to do during the authentication process is to project the Encrypted Query Image into FaceSpace(low dimension space). The projection process will not be straightforward because we are working in the encrypted domain.

Suppose, Encrypted query image is denoted by $[W_{[1xd]}]$. This is an input for the server. Now, the server has to perform projection to make an encrypted feature vector query image. As we are working in a cryptography environment, we have to use the homomorphic properties of Paillier the algorithm to do operations on the encrypted image. Now, we will go through the projection of the algorithm in the encrypted domain. Firstly, we have to find the mean subtracted query image. We have a mean vector $M_{[1xd]}$ and also FaceSpace consists of eigenvector $Y_{[Dxd]} = [y_1, y_2, y_3 \ . \ . \ . \ y_d]$ from the previous section of PCA.

$$\gamma_{[1xD]} = W_{[1xD]} - M_{[1xD]} = \begin{pmatrix} W_1 \,.\, m_1 \\ W_2 \,.\, m_2 \\ . \\ . \\ . \\ W_D \,.\, m_D \end{pmatrix} \tag{13}$$

The next step is to find each feature $u_{i[1x1]}$] by projecting Image $\gamma_{[1xD]}$ onto eigenface$y_{i[1xD]}$ So, Dot product of $\gamma_{[1xD]}$ and $y_{i[1xD]}$ will give $u_{i[1x1]}$.

$$u_i = (y_{i1}, y_{i2}, y_{i3} \ . \ . \ . \ y_{iD})_{[1xD]} \cdot \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ . \\ . \\ . \\ \gamma_D \end{pmatrix}_{[Dx1]} \tag{14}$$

$$u_i = \gamma_1 . y_{i1} + \gamma_2 . y_{i2} + \gamma_3 . y_{i3} + \ . \ . \ . \gamma_D . y_{iD} \tag{15}$$

$\forall$ i $\in$ {1, 2, 3 . . . d}

But the Server has an encrypted query image [W]. So, we cannot directly apply the above formulas and find u_i (feature vector). To find an encrypted feature vector, the Server must apply the Homomorphic properties of the Paillier Cryptosystem. (Refer 2. Encryption Module for Homomorphic properties.)

Using homomorphic properties, the Server will be able to find encrypted mean subtracted query image as,

$$[\gamma] = [W - M] = \begin{pmatrix} [W_1] . [m_1] \\ [W_2] . [m_2] \\ . \\ . \\ . \\ [W_D] . [m_D] \end{pmatrix} = \begin{pmatrix} [W_1] . [m_1]^{-1} \\ [W_2] . [m_2]^{-1} \\ . \\ . \\ . \\ [W_D] . [m_D]^{-1} \end{pmatrix} \tag{16}$$

The server has $M = [m_1, m, m_3 \ . \ . \ . \ m_D]$ in plain form. So, the server will find encrypted mean values that are [− M].

Now, the server can find encrypted mean subtracted query image $\gamma_{[1xD]}$. Since the mean subtracted query image is encrypted, it can only find encrypted query feature vector using homomorphic properties and the formula of u_i that we have written above.

$$[u_i] = [\gamma_1 . y_{i1} + \gamma_2 . y_{i2} + \gamma_3 . y_{i3} + \ . \ . \ . \gamma_D . y_{iD}] \tag{17}$$

Applying homomorphic property on addition,

$$[u_i] = \prod_{j=1}^{D} [\gamma_i . y_{ij}] \tag{18}$$

Applying homomorphic property on multiplication,

$[u_i] = \prod_{j=1}^{D} [\gamma_i]^{y_{ij}}$ (19)

$\forall$ i $\in$ {1, 2, , 3 . . . d}

The server has transformation matrix $Y_{[Dxd]} = [y_1, y_2, y_3 \ . \ . \ . \ y_d]$ in plain form. So, Finally, we have an encrypted query feature vector $u_{[1xd]} = [u_1, u_2, u_3 \ . \ . \ . \ u_d]$. Important note, When we are applying homomorphic property we have to be very careful about which variable is encrypted which is in plain form.

The server will use this procedure to find encrypted query feature vectors during the authentication process. Only after finding the encrypted query feature vector, the server can proceed to do another task in the authentication process.

IV. MATCHING MODULE

In this module, we will have the user's encrypted feature vector [u] as an input coming from the second module of Encryption. Other than that, we have N templates $X = [x_1, x_2, x_3 \ . \ . \ . \ x_N]$ stored into the server. These templates also feature vectors having d features and hence of dimension 1xd. We have divided this module into two sub-modules.

A. Distance Calculations

In this submodule, we will be computing the Euclidean distance between the user's queried feature vector and the templates stored on the server. But as we discussed earlier we cannot use the actual data neither we can decrypt the data during the authentication. Here the homomorphic property of the Paillier scheme will come in aid.

Here the user vector is $u = \{u_1, u_2, u_3 \ . \ . \ . \ u_d\}$ and the i^{th} template is $x_i = \{x_{i1}, x_{i2}, x_{i3} \ . \ . \ . \ x_{id}\}$. Hence the Euclidean distance between these two is given by,

$$EUD(u, x_i) = \sqrt{\sum_{j=1}^{d} (u_j - x_{ij})^2} \quad (20)$$

Where EUD is the notation we will be using for Euclidean distance. Here, as we just have to do a comparison between these distances, we don't need to care about the square root, we can just use squared Euclidean distance for simplicity.

$$EUD(u, x_i)^2 = \sum_{j=1}^{d} (u_j - x_{ij})^2 \quad (21)$$

$$= \sum_{j=1}^{d} u_j{}^2 + \sum_{j=1}^{d} x_{ij}{}^2 - \sum_{j=1}^{d} 2u_j x_{ij} \quad (22)$$

Here in the summation, there are three terms. But here at the server, we don't have the values of u_j. All that we have got is an encrypted feature vector [u]. Here we will use the additive homomorphic property which tells that decryption of multiplication of ciphertext gives the same value as the addition of corresponding actual plain texts. Here if I denote these three terms as term1, term2 and term3 then,

$EUD^2(u, x_i) \ mod \ n = D \ (E(term_1) \ E(term_2) \ E(term_3) \ mod \ n^2)$

$here \ term_1 = \sum_{j=1}^{d} u_j{}^2, term_2 = \sum_{j=1}^{d} x_{ij}{}^2 \ and \ term_3 = \sum_{j=1}^{d} 2u_j x_{ij}$

Here we will be using the E(.) and [.] notations interchangeably to show encryption. Now, as the server has the values of x_{ij}, $E(term_2)$ can be computed at the server. Encryption of $term_3$ can also be computed again using the homomorphic properties,

$$E\left(\sum_{j=1}^{d} -2u_j x_{ij}\right) = \prod_{j=1}^{d} E\left(-2u_j x_{ij}\right) \quad (23)$$

$$E\left(-2u_j x_{ij}\right) = E(u_j)^{-2x_{ij}} \quad (24)$$

The second expression is from the second property mentioned in the encryption module stating that encryption of multiplication of two plain texts is the same as anyone ciphertext raised to the power of plain text of other ciphertexts. Here, as the server knows the value of x_{ij} j (plain text), we can take it multiplied by -2 as the power. But the encryption of the first term is not possible at the server end. So the server will have to take the help of the client for this encryption. So the server will send these [u] to a client who will first decrypt it, they will take some of the squares of each element and will encrypt it and send it back to the server. But there is a problem. We can not just send the feature vector to the client because from that, he can get the information about the transformation matrix of the server, which includes some information about the templates stored on the server. So rather than sending the encrypted vector straight away, he will first try to add some randomness to it.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue III Mar 2022- Available at www.ijraset.com

The server will choose d random variables r_j for $1 \leq j \leq d$, and will add this random variable to the feature vector using the homomorphic property. That is, $z_j = u_j + r_j$ and hence, $E(z_j) = E(u_j)\, E(r_j)$. Then, he will send this encrypted random feature vector [z] to the client. The client will decrypt it and will send back the encryption of summation of square of z_j let's say we call it $term_3' = \sum_{j=1}^{d} z_j^2$.

To get $term_3$ from this, we may notice that,

$$u_j^2 = (u_j + r_j)^2 - r_j^2 - 2u_j r_j$$

$$= x_j{}^2 - r_j^2 - 2u_j r_j \quad (25)$$

Hence using the homomorphic properties.

$$E(u_j^2) = E(x_j{}^2)E(r_j^2)E(-2u_j r_j)$$

$$= E(x_j{}^2)E(r_j^2)E(u_j)^{-2r_j} \quad (26)$$

Here at some of the expressions, we have neglected the term mod n because n is very huge compared to any of the values to which we are taking modulo. So it just will not make any difference of having or not having mod n. So from the above expressions, we can get the $E(term_3)$ as follow,

$$E(term_3) = E\left(\sum_{j=1}^{d} u_j^2\right) \quad (27)$$

$$= \prod_{j=1}^{d} E(u_j^2)$$

$$= \prod_{j=1}^{d} E(x_j{}^2)E(r_j^2)E(u_j)^{-2r_j} \quad (28)$$

$$= E(term_3') \prod_{j=1}^{d} E(r_j^2)E(u_j)^{-2r_j} \quad (29)$$

Now the server knows the value of $E(term_3')$, r_j and $E(u_j)$. Hence, the value of $E(EUD^2(u, x_i))$ can be computed. This encrypted value will be used to find the minimum distance between them.

B. Minimum Distance

In this module, we will be finding the minimum distance among the distances calculated in the first submodule. Here, again the restriction is the same, that we can not decrypt the distance. Because if the server decrypts the distance, then one can get the details about the user's feature vector as well as about the templates of the server. At the same time, we can not compare two encrypted values, because there is randomness in every encrypted value which gives the wrong comparison of two entities.

To do the comparison such that all the requirements get satisfied, we again have to take the help of the client. Before that, we may notice that, in the first homomorphic property mentioned in the encryption module, we are taking modulo n sum on the left-hand side of the expression.

Now, consider that the result of this addition is negative but let's say that the absolute value of the addition is less than n/2. Let's say that these two numbers are a and b such that,

$$a + b < 0 \quad (30)$$

$$abs(a + b) < \frac{n}{2} \quad (31)$$

Where abs(.) is an operator giving absolute value. Now if we take the modulo of this addition then,

$$a + b \bmod n = n - abs(a + b) \bmod n$$
$$= n - abs(a + b) \leq \frac{n}{2} \quad (32)$$

So, we can say that, if it is given that abs(a+b) < $\frac{n}{2}$, then if a + b < 0, the modulo n will be more than $\frac{n}{2}$ 2 otherwise it will be less than $\frac{n}{2}$. The first condition of addition having less than $\frac{n}{2}$ will be satisfied in our case because of the reason that n is very big (usually 1024 bit ≈ 309 decimal digits long) compare to any plain text that we are encrypting throughout our protocol.

Now coming back to the comparison part, let's say there are two distances d_a and d_b and we want to check if $d_a < d_b$ or not. For that, we will look at the subtraction of two numbers. If $d_a - d_b < 0\ then\ d_a < d_b$ otherwise $d_a \geq d_b$. But here, as we don't know the actual value of d_a and d_b, rather we know $E(d_a)$ and $E(d_a)$, we have to use the homomorphic properties to get encryption of the subtraction of two distances that is,

$$d_a - d_b \bmod n = D(E(d_a)\ E(d_b)^{-1} \bmod n^2) \quad (33)$$

Now as the server can not decrypt anything, he will have to send it to the client. But again as he can not expose the transformation matrix, he has to randomize this number so that after decrypting it, the client will not get any information from it. To randomize it, we will multiply this subtraction with some random positive integer r such that the sign of the subtraction does not get affected and the multiplication satisfies the condition of being less than $\frac{n}{2}$.

$$r\ .\ abs(da - db) < \frac{n}{2} \quad (34)$$

In encryption domain it will be,

$$r \cdot (d_a - d_b) \bmod n = D((E(d_a)\ E(d_b)^{-1})^r \bmod n^2) \quad (35)$$

The server will compute the value on the right-hand side of the above expression and will send it to the client for decryption. The client will decrypt it and will check the value of decryption. If the value is less than ^{n} then the client will send back the message to the server that d_a,d_b otherwise $d_a < d_a$. This is how the server will know the minimum between d_a and d_b. So, the server will first initialize one variable let's say D_{min} which will be the minimum distance for the queried user feature vector. It will iterate through all the N templates and every time during the i_{th} iteration, it will do all the above processes to get the minimum between the current minimum D_{min} and the distance from i^{th} iteration d_i. Based on this value D_{min} the server will decide, in the decision module, whether the user is valid or not

V. DECISION MODULE

The decision module is the last module of this problem domain. **The objective of this module** is to tell whether the user is authenticated or not. This module will use the output of the Matching module as an input. The output of the matching module is the minimum distance between queried feature vector and templates from the server database. Comparatively, this is a small module. Now, we will look into the the algorithm that will decide "Is this the user is authenticated or not"?

1) Algorithm 2: The algorithm to decide the authenticity of the user

Input-

1. Minimum distance(D_{min}) between the user-queried feature vector and templates from the server's database.
2. Pre-define threshold(γ). This threshold value is not deterministic.

Output- if The user is authenticated return 1 else return 0. Basically, output is a boolean value.

Procedure-

1: if $D_{min} \leq \gamma$ then
2: return 1 (The user is genuine) <
3: else
4: return 0 (The user is the intruder)
5: end if

Now, we will see the method to find γ Threshold value. Although there is no specific algorithm to find threshold value in the authentication process. So, we have to find thresholds using statistical methods such that accuracy will be as high as possible. This is a pre-authentication process. So, we can only use data that is stored on the server.

2) The Algorithm 3: Method to find threshold γ.
Input- Feature vectors that are stored in the server database. Let's assume $V_{[Kxd]} = [\ v_1, v_2, v_3, \ldots, v_K\ ]$ is vector of K feature vector. Each feature vector is of d-dimension.

Procedure-
1: $Max_{global} \leftarrow INT\_MIN$
2: for $i = 1\ ;\ i \leq K\ ;\ i++$ do
3: $min_{local} \leftarrow INT\_MAX$
4: for $j = 1\ ;\ j \leq K\ ;\ j++$ do
5: if $i == j$ then
continue;
6: end if
7: $dist \leftarrow 0$
8: for $l = 1\ ;\ l \leq d\ ;\ l++$ do
$dist\ +=\ (v[i,l] - v[j,l])^2$
9: end for
10: $min_{local} \leftarrow min(min_{local}, dist)$
11: end for
12: $Max_{global} \leftarrow max(Max_{global}, min_{local})$
13: end for
14: Threshold γ ← Max_{global}

Here, we can see that the threshold value decides the outcome whether the user is genuine or not. So, it is a very important parameter. We have to be very careful when we are choosing γ. And two kinds of errors can occur during the authentication process.

- False Acceptance (FA): Occurs when the authentication process allows an intruder(who hasn't enrolled) into the system.
- False Rejection (FR): Occurs when the authentication process rejects a genuine user(who has enrolled in the system).

We have to reduce FA, FR by choosing the appropriate threshold γ. Though at some extent we can tolerate FR(False Rejection) because in that case, the user can try again to enter into the system. But we cannot tolerate FA(False Acceptance) because this will lead to very dangerous consequences as the authentication process allows intruders into the system.

Now we will see how accurate our authentication process is. There is a formula to find the accuracy of the authentication system using FA and FR.

$$Accuracy(\%) = \left\{100 - \frac{[FAR(\%)+FRR(\%)]}{2}\right\} \quad (36)$$

Here, FAR is False Acceptance Rate. The formula for FAR is written below

$$FAR = \frac{Total\ number\ of\ FA}{Total\ number\ of\ intruder} \quad (37)$$

FAR is False e Rejection Rate. The formula for FRR is written below

$$FRR = \frac{Total\ number\ of\ FR}{Total\ number\ of\ genuine} \quad (38)$$

VI. ANALYSIS

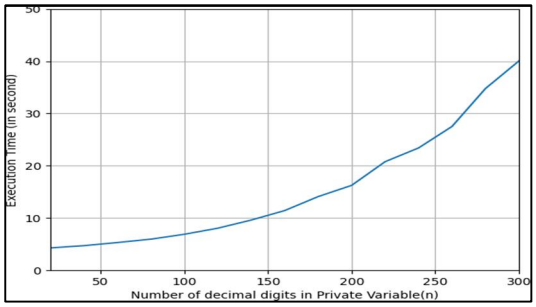


Figure 2: A varying number of decimal digits of n = p*q (Private Parameter) Parillier

From the above graph, we can observe that as we increase the number of decimal digits in n, execution time is also increasing. But there is one thing we can see that till 150 decimal digits there is not much change in execution time(from 5 sec to 10 sec) and it implies that the slope of the curve is small. But as we increase the number of digits from 150 to 300 slope of the curve is becoming larger and larger, which implies that a small change in some digit time will cause a large change in execution time.

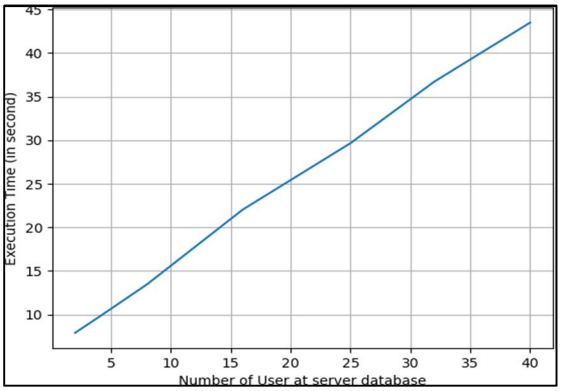


Figure 3: A varying number of templates(the users) in the server database

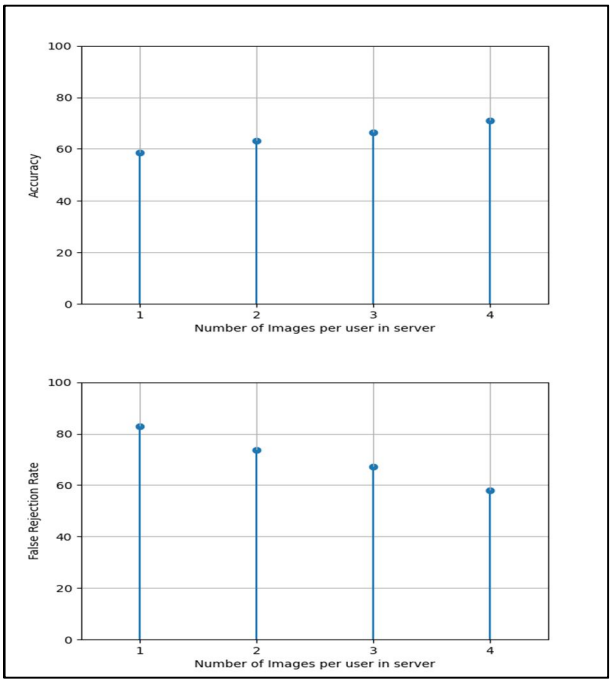


Figure 4: A varying number of images per person in the server database

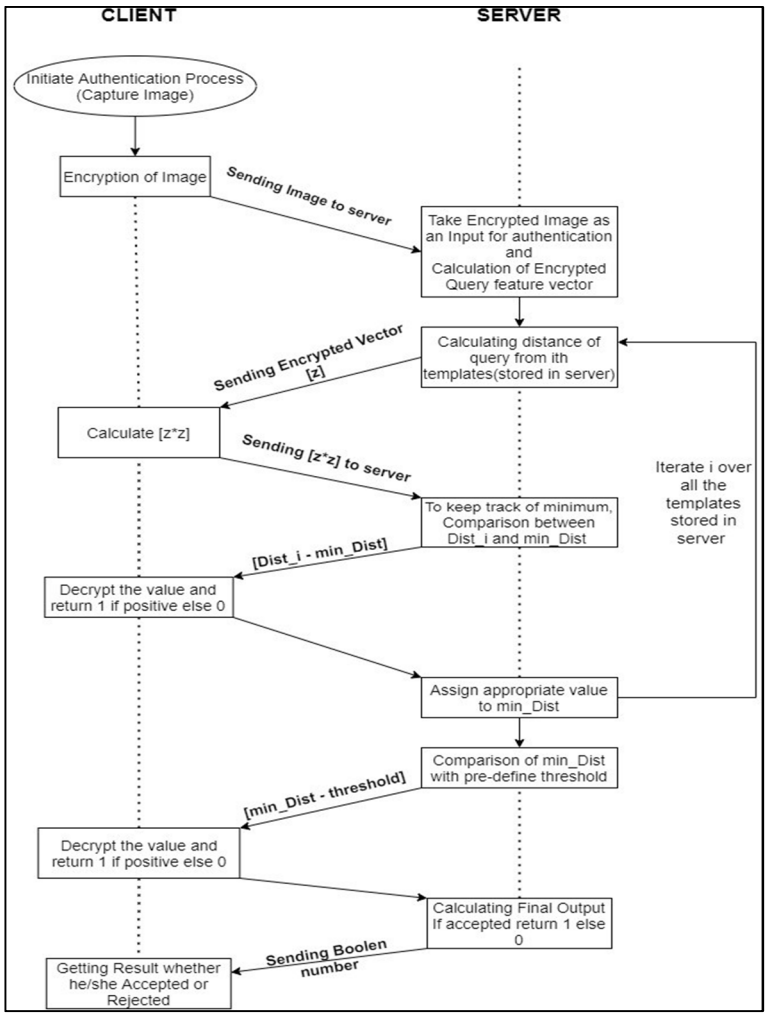
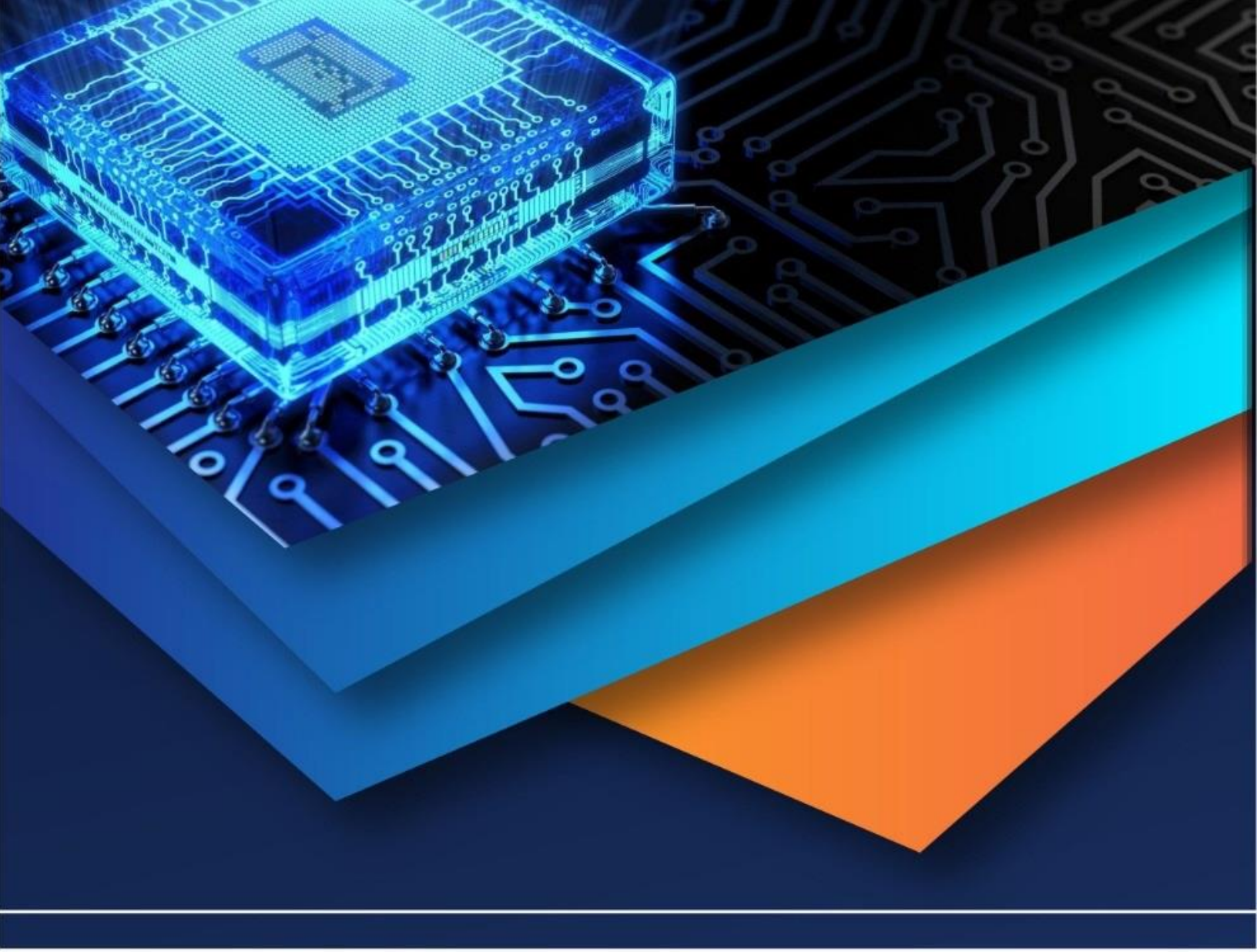


Figure 5: 2-way secure communication model for the Authentication process

REFERENCES

[1] Paillier, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
[2] H. Chun et al., Outsourceable Two-Party Privacy-Preserving Biometric Authentication , Proc. 9th ACM Symp. Info. Computer and Commun. Security, pp. 401-12, June 2014.
[3] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving Face Recognition. , In 9th International Symposium on Privacy Enhancing Technologies, 2009.
[4] Malik Jyoti, Girdhar Dhiraj, Dahiya, Ratna Sainarayanan, I. Lagendijk and T. Toft. Reference Threshold Calculation for Biometric Authentication. , International Journal of Image, Graphics and Signal Processing.
[5] R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. Smith, “Reusable fuzzy extractors for low-entropy distributions,” in Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 117–146, Springer, 2016.
[6] A. I. Desoky, H. A. Ali, and N. B. Abdel-Hamid, “Enhancing iris recognition system performance using templates fusion,” Ain Shams Engineering Journal, vol. 3, no. 2, pp. 133–140, 2012.
[7] K. P. Hollingsworth, K. W. Bowyer, and P. J. Flynn, “Improved iris recognition through fusion of hamming distance and fragile bit distance,” IEEE transactions on pattern analysis and machine intelligence, vol. 33, no. 12, pp. 2465–2476, 2011.
[8] N. K. Ratha, J. H. Connell, and R. M. Bolle, “Enhancing security and privacy in biometrics-based authentication systems,” IBM systems Journal, vol. 40, no. 3, pp. 614–634, 2001.
[9] A. T. B. Jin, D. N. C. Ling, and A. Goh, “Biohashing: two factor authentication featuring fingerprint data and tokenised random number,” Pattern recognition, vol. 37, no. 11, pp. 2245–2255, 2004.
[10] R. Ang, R. Safavi-Naini, and L. McAven, “Cancelable key-based fingerprint templates,” in Australasian conference on information security and privacy, pp. 242– 252, Springer, 2005.
[11] S. Kanade, D. Petrovska-Delacr´etaz, and B. Dorizzi, “Cancelable iris biometrics and using error correcting codes to reduce variability in biometric data,” in Comuputer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 120–127, IEEE, 2009.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY