CHAPTER 5

DESIGN AND IMPLEMENTATION OF THE PROPOSED SYSTEM

5.1 PCASA - SYSTEM OVERVIEW

The Principal Component Analysis with signcryption algorithm has the following steps:

Step 0: Start.
Step 1: Capture video.
Step 2: Extract images from video.
Step 3: Generate eigen faces.
Step 4: Get signature from the user.
Step 5: Encrypt sign image.
Step 6: Generate DCS.
Step 7: Extract details.
Step 8: Match data.
Step 9: If matched, allow entering.
Step 10: else do step1.
Step 11: Update trained set.
Step 12: Stop.
The user’s personal data have been entered at the time of the registration by the PCASA system. Different postures of clipping have been stored into the system when they are entered into it. At first time, the user’s signature is gathered. The encrypted signature image is fed into the cover media. The system will generate the DSC (Dynamic Secret Code) for the user. This secret code is used by the users whenever they enter into the system again. DSC extracts all the details relating to the user registered already. Whenever a user enters into the system, his/her face will be recognized by the system with the DSC. If the data matched, they can enter into the system successfully. Otherwise, the system is required to verify the signature by other methods. If all data matching is failed, user will be rejected.

5.2 SYSTEM ARCHITECTURE DESIGN

![Figure 5.1 PCASA stages](image-url)
Figure 5.1 illustrates the block diagram of the proposed system.

New user: The person who enters into the system is considered as a new user.

Capture video: System has a video capturing tool to capture the face. This tool automatically captures the user’s activities. Therefore, different postures are captured and fragmented as a series of images.

Get signature: User’s signature has been obtained to embed.

Collect different postures of face: In order to recognize the face of the user, we need several postures of the face are needed. These have been stored into the data base.

Update images: Video frames are fragmented into a set of images and stored into the database.

Feature extraction: The principal components of the user’s face are extracted using eigen vectors.

Generate signcryption image: A signcryption image has been generated using PCASA.

Generate DSC: Dynamic Secret Code has been generated by the system for authentication.

Validate the user: The right person alone can be entered into the system after checking. The embedded processes are:

Face detection: User face has been recognized by the PCASA method.
Compare with database images: Whenever a user enters the system, his/her face is compared with the database images.

Match data: The user’s data has been matched by DSC. When matched, any person can enter.

All these processes are explained in the following sections.

5.3 VIDEO CAPTURING

![Figure 5.2 Sample video clipping](image)

This is the process of capturing the video clip of the user when he/she enters into the system to detect the face. To avoid the camera attaching problem, we provide both the manual input getting method and automatic capturing method. MPEG-4, AVI file formats are tested for this process. Figure 5.2 illustrates the captured video sample.
5.4 EXTRACTING IMAGES FROM VIDEO

![Sample of extracted images from video clipping](image)

Figure 5.3 Sample of extracted images from video clipping

The images of Figure 5.3 have been extracted from the sample video of Figure 5.2. Different expressions have been expressed by the user. This trained set has been stored at the back end folder.

5.5 APPLYING PCASA

A signcrypted image has been created with the use of a system generated DSC and trained sample data. PCASA is the method to generate signcrypted image content to verify a user.

Sample images are used for user authentication. The user is asked to enter his/her video clipping either manually or using video generation method. These clippings have been divided into several postures of the images as shown in Figure 5.3. Then the signature of the user has been gathered either manually or automatically. Figure 5.4 shows the sample user’s face with his/her signature needed to apply PCASA.
Figure 5.4 Sample users with his/her signatures

5.6 FEATURE EXTRACTION

Figure 5.5 Feature extraction of the user
Figure 5.5 shows the feature extraction of the given image. To recognize a user’s face, the features of the individual have to be obtained as follows:

### 5.6.1 Principal Component Analysis

Let there be \( R \) face images in the training set, where each image \( X_i \) is a two dimensional array of size \( m \times n \) of intensity values. The image \( X_i \) can be converted into a vector of \( D \) (where \( D = m \times n \)) pixels. The rows of pixels of the image are placed one after another to form a vector.

If the training set of \( R \) images is defined by \( X=(X_1, X_2, X_3, ..., X_R) \), then the covariance matrix is defined as:

\[
\Gamma = \frac{1}{R} \sum_{i=1}^{R} (X_i - \bar{X})(X_i - \bar{X})^T = \Phi \Phi^T
\]  

(5.1)

Where \( \Phi = (\Phi_1, \Phi_2, \Phi_3..., \Phi_R) \subset \mathbb{R}^{D \times R} \) and

\[
\bar{X} = \frac{1}{R} \sum_{i=1}^{R} X_i
\]  

(5.2)

is the mean image of the training set. Also, the dimension of the covariance matrix is \( D \times D \).

The eigen values and eigen vectors are then calculated from the covariance matrix.

Let \( Q= (Q_1, Q_2, ..., Q_R) \subset \mathbb{R}^{D \times R} \) (generally, \( r < R \)) be the \( r \) normalized eigen vectors corresponding to \( r \) largest eigenvalues. Each of the \( r \) eigen vectors is called an eigenface. Now, each of the face images of the training set \( X \) is projected into the eigenface space to obtain its corresponding eigenface based feature \( Z_i \subset \mathbb{R}^{D \times R} \) which is defined as:

\[
Z_i = QTY_i \quad i=1, 2 ... R
\]  

(5.3)

where \( Y_i \) is the mean-subtracted image of \( X_i \).
5.6.2 Classifications of an Image

Classification includes image sensors, image pre-processing, object detection, object segmentation, feature extraction and object classification. Some of the techniques to classify the images are Artificial Neural Network (ANN), Decision Tree (DT), Support Vector Machine (SVM), Principal Component Analysis (PCA).

Principal Component Analysis (PCA) is implemented from applied Linear Algebra. PCA is used for analysis from Artificial intelligence to computer vision, because of simplicity, extractness of relevant information. PCA provides a route to achieve the spatial features without any complication.

5.6.3 PCA: Eigen vectors of covariance

Researchers derive an algebraic solution to PCA using Linear Algebra. This solution is based on an important property of eigen vector decomposition. The data set is $X$ which is an $m \times n$ matrix, where $m$ is the number of measurement types and $n$ is the number of samples. The goal is summarized as follows:

An orthonormal matrix $P$ is found such that $Y = PX$ and $CY \equiv \frac{1}{n-1} \times YPY^T$ is diagonalized. The rows of $P$ are the principal components of $X$. We begin by rewriting $CY$ in terms of our variable of choice $P$.

$$CY \equiv \frac{1}{n-1} \times YP^T$$  \hspace{1cm} (5.4)

$$CY \equiv \frac{1}{n-1} \times (PX)(PX)^T$$  \hspace{1cm} (5.5)

$$CY \equiv \frac{1}{n-1} \times PX^T P^T$$  \hspace{1cm} (5.6)
It has been defined as a new matrix $A = XX^T$, where $A$ is symmetric. The roadmap is to recognize that a symmetric matrix ($A$) is diagonalized by an orthogonal matrix of its eigen vectors. For a symmetric matrix:

$$A = EDE^T$$ \hfill (5.9)  

Where $D$ is a diagonal matrix and $E$ is a matrix of eigen vectors of $A$ arranged as columns. The matrix $A$ has $r \leq m$ orthonormal eigen vectors where $r$ is the rank of the matrix. The rank of $A$ is less than $m$ when $A$ is degenerate or all data occupy a subspace of dimension $r \leq m$ while maintaining the constraint of orthogonally. This situation results by selecting $(m - r)$ additional orthonormal vectors to make the matrix $E$. These additional vectors do not affect the final solution because the variances associated with these directions are zero. We select the matrix $P$ to be a matrix where each rows of $P$ is an eigen vector of $XX^T$, by this selection $P = E^T$. Substituting into equation (5.10), we can find the value $A = P^TDP$. With this relation ($P^{-1} = P^T$), by evaluating $CY$, as follows:

$$CY \equiv \frac{1}{n-1} \times PAP^T \hfill (5.10)$$

or

$$CY \equiv \frac{1}{n-1} \times P(P^TDP)P^T \hfill (5.11)$$

or

$$CY \equiv \frac{1}{n-1} \times P(XX^T)P^T \hfill (5.7)$$

$$CY \equiv \frac{1}{n-1} \times PAP^T \hfill (5.8)$$
\[ CY \equiv \frac{1}{n-1} \times (PP^T)D(PP^T) \] (5.12)

or

\[ CY \equiv \frac{1}{n-1} \times (PP^I)D(PP^I) \] (5.13)

or

\[ CY \equiv \frac{1}{n-1} \times (I)D(I) \] (5.14)

It is evident that the choice of \( P \) diagonalizes \( CY \). This is the goal for PCA.

### 5.6.4 Eigen faces

The Eigen Face Method is the simplest method to recognize the human face. The fundamental concept of an eigen face is information reduction. An evaluated image may contain a lot of information about the image. In order to retrieve the important features of face image, the sub space area is calculated. Base images are gathered from by given human face image in different poses or angles and stored under a trained set. Standard classification methods are used to classify those images under some category by their features. The features have to be extracted after finding the face space and should be represented as a vector. There are several methods for classification of the images such as k-nearest neighbor approach, neural network, Euclidean distance measurement.

The steps indicated below are followed by PCA:

i) Generation of eigen face

ii) Detection of face space among image space

iii) Extraction of features

iv) Comparison with the trained set of image features.
5.6.5 Eigen vectors

Sample faces are needed, before any work can be done to generate the eigen faces. These images will be used as examples of what an image in face-space looks like. They do not necessarily need to be images of the people the system would later use for identification (though it can help); however, the image should represent variations one would expect to see in the data on which the system is expected to be used, such as head tilt/angle, a variety of shading conditions, etc. Ideally, these images should contain pictures of faces at close to the same scale, although this can be accomplished through preprocessing if necessary. All of the images used in the system, both sample and test images, are required to be of the same size. The resulting eigenfaces will also be of this same size once they have been calculated.

All images being dealt with are assumed to be Gray scale images, with pixel intensity values ranging from 0 to 255. Suppose, there are $k$ images in our data set. Each sample image will be referred to as $X_i$ where $i$ indicate number of samples images used, ($1 \leq i \leq k$). Each $X_i$ is a column vector.

Generally, images are thought of as pixels, each having $(x, y)$ coordinates with $(0, 0)$ being at the upper left corner (or one could think of an image as a matrix with $y$ rows and $x$ columns). Converting this to a column form is a matter of convenience and can be done in either a column or a row major form. As long as it is done consistently for all sample images, the outcome will not be affected. The size of the resulting $X_i$ column vector will depend on the size of the sample images. If the sample images are $x$ pixels across and $y$ pixels tall, the column vector will be of size $(x \times y) \times 1$. These original image sizes should be remembered if one wishes to view the resulting eigenfaces, or projections of test images into face-space. This is to allow a normal image to be constructed from a column vector of image pixels. Let $\overline{X}$
be the mean of all $X_i (1 \leq i \leq k)$. This is the step for calculation of an average face of the database. If the interpretation of the vector is done as a normal image, the next step is calculation of the difference faces $U_i$ such that $U_i = X_i - X_\text{mean}$, (where $X_\text{mean}$ is mean) and form a matrix $U$, such that $U = [U_1, U_2, \ldots, U_k]$. The goal now is to generate the eigenfaces and this is done by calculating the eigen vectors of the covariance matrix $UU^T$. This cannot be done directly. The size of $UU^T$ is $(x * y)*(x * y)$ which is very large. Clearly, doing these calculations on a resulting matrix of this size is going to be taxing on all but the most specialized, advanced hardware. To avoid this problem, a trick from Linear Algebra is applied.

The eigen vectors of the $UU^T$ matrix can actually be found by considering linear combinations of the eigen vectors of the $U^TU$ matrix. This is extremely useful when it could be the size of the $U^TU$ matrix is realized as $k \times k$, where, in real world, situations $k \ll (x*y)$. The eigen vectors $w_j$ of this matrix can be readily found through the following formula:

$$W_j = \frac{\sum_{i=1}^{k} U_i E_{ij}}{\sqrt{\lambda_j}}$$

(5.15)

Where $E_{ij}$ is the $l$th value of the $j$th eigen vector of $U^TU$ and $\lambda_j$ is the corresponding eigen value of $W_j$ and $E_j$. The Linear Algebra part of this trick is given below:

Let the eigen vectors of $U^TU$ be $E_j (1 \leq j \leq k)$ and the corresponding eigen values be $\lambda_j$. Hence, we can write:

$$U^TU E_j = \lambda_j E_j$$

(5.16)

Premultiplying both the sides by $U$:

$$U \times U^TE_j = \lambda_j UE_j$$

(5.17)
Thus, \(w_j = UE_j\) is the \(j\)th eigen vector of \(UU^T\) with corresponding eigen value \(\lambda_j\). The fact is that the eigen values for the \(UU^T\) and \(U^TU\) are the same (though if we were going to calculate all of the eigen values of the \(UU^T\) matrix, we could get more values, the eigen vectors of the \(U^TU\) only represent the most important subset of the eigen values of the \(UU^T\) matrix).

5.6.6 Generation of stegano image

The generated eigen faces will be used for making stegano image. User signature image (\(S_n\)) has been fragmented into \(S_1, S_2 \ldots S_n\) are considered as subsets of \(S\). Fragmented signature is embedded within the trained set of images. Sum of \(S\) can be divided into several parts such as \(S_1, S_2 \ldots S_n\).

\[\sum_{i,j=1}^{n} S = s_1 \cup s_2 \cup \ldots \cup s_n, \text{ where } n = \text{number of images}.\]

5.6.7 Generation of dynamic secret code

User personal details have been registered at the time of login. The data gathered from the user are used to generate the Dynamic Secret Code (DSC). It is the arrangement order of fragmented stegano image. This is used as the key to enter into the system. The system will renew the code everytime. Steps to generate dynamic secret code are:

Step 1: Start
Step 2: Assign order \(d\) for \(S_n\)
Step 3: Temp= jumble \((S_n, d)\)
Step 4: Renew \(d\).
Step 5: Publish it to user.
Step 6: Stop
5.7 MATCHING WITH ORIGINAL DATA

Different user scenarios have been explained in Chapter 4. The system will react for the given input depending on the type of the user.

The user should give his/her dynamic secret code to the system which has been already given to them. DSC has the order of the fragmented stegano image. The system captures the user’s activities for one second. These video clippings will be converted into a set of images. These images have been matched with the existing one by using DSC.

PCASA will compare the encrypted video with the captured one. If the given data have been matched, PCASA allows the user entry into the system. According to the eigen vector and order of fragmented signature the image is denoted as d, the user data will be matched. If the data do not matched with the existing one, user will be rejected. The function DSC is derived as $O(S) = d$, where $S$ is the user signature image.

5.8 IMPLEMENTATION OF DIFFERENT STEGANO ALGORITHMS

Several encryption techniques are available. The following algorithms have been implemented with the signcryption to compare the experimental results:

5.8.1 SHA-256 algorithm

Hash value of the given data has been created by using SHA-256. It is used to create the fixed size key for any length of source data. The advantage of SHA-256 is that it never shows the source data. It is a one way
encryption mechanism. Length of the encrypted hash value is the disadvantage.

5.8.2 DWT with Mallat algorithm

Mallat (1998) algorithm is used to implement fastest way to compute continuous time wavelet expansion:

1. Compute $\hat{a}$ from $j$ to $J$ at finest resolution $J$
2. Recursively compute $\hat{a}$ from $j$ to $i$ and compute $a$ from $j$ to $i$

Recursive function of bigger $i \rightarrow$ coarser, until we get finer $\rightarrow$ coarser.

This algorithm does purely discrete-time processing, since wavelet coefficients can be produced very quickly.

DWT (Discrete Wavelet Transform) has been used to get faster output.

5.8.3 MD5 algorithm

Message digest has been created for a given data. As also hash value of the given data. Identical size key generation and multiple hashing with recursive procedure is the difficulty of the algorithm. MD5 used to generate message digest for fixed size of key.

5.8.4 Swapping algorithm

Swapping algorithm enables simple encryption method. 32 bit data has been used for the encryption. Simple AND, XOR functions are being used. It produced quick encryption than other algorithms.
5.8.5 **Blowfish algorithm**

One of the easily available algorithms is the Blowfish algorithm. It uses two important mathematical functions to create a key. It is useful for plaintext encryption. Blowfish encryption used addition, XOR functions.

\[
L_0 \& R_0 \text{ for } i = 1 \text{ to } 16 \text{ do}
\]

\[
R_i = L_{i-1} \text{ XOR } P_i;
\]

\[
L_i = F[R_i] \text{ XOR } R_{i-1}; \quad L_{17} = R_{16} \text{ XOR } P_{18};
\]

\[
R_{17} = L_{16} \text{ XOR } i_{17};
\]

where \( F[a,b,c,d] = ((S_1,a \text{ XOR } S_2,b) \text{ XOR } S_3,c) \text{ XOR } S_4,d \)

This method has been used to get plain text output.

5.9 **CONCLUSIONS**

Chapter 5 explains the merits and demerits of some of the existing encryption methodologies such as swapping, MD5, SHA. The methodologies have been carried out by us using the conventional and nonconventional key. Mechanism of encryption and decryption involves both transposition and substitution.