VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A PROJECT REPORT **(15CSP85)** ON

"BIOMETRIC PRIVACY"

Submitted in Partial fulfillment of the Requirements for the Degree of

Bachelor of Engineering in Computer Science & Engineering

By

K.H MEGHANA (1CR16CS062)

KANIKA (1CR16CS066)

MONISHA.G (1CR16CS093)

SUSHMITHA B.V (1CR16CS169)

Under the Guidance of,

SHIVARAJ V B

Assistant Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled **"BIOMETRIC PRIVACY"** carried out by **Ms. K.H MEGHANA**, **1CR16CS062**, **Ms. KANIKA**, **1CR16CS066**, **Ms. MONISHA.G**, **1CR16CS093, Ms. SUSHMITHA B.V**, **1CR16CS169,** bonafide students of CMR Institute of Technology, in partial fulfillment for the award of **Bachelor of Engineering** in Computer Science and Engineering of the Visveswaraiah Technological University, Belgaum during the year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Shivaraj V B	**Dr. Prem Kumar Ramesh**	**Dr. Sanjay Jain**
Assistant Professor	**Professor & Head**	**Principal**
Dept. of CSE, CMRIT	**Dept. of CSE, CMRIT**	**CMRIT**

External Viva

Name of the examiners — Signature with date

1. _____

2. _____

DECLARATION

We, the students of Computer Science and Engineering, CMR Institute of Technology, Bangalore declare that the work entitled "**BIOMETRIC PRIVACY**" has been successfully completed under the guidance of Prof. Shivaraj V B, Computer Science and Engineering Department, CMR Institute of technology, Bangalore. This dissertation work is submitted in partial fulfillment of the requirements for the award of Degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2019 - 2020. Further the matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place:

Date:

Team members:

K.H MEGHANA (1CR16CS062) _____

KANIKA (1CR16CS066) _____

MONISHA.G (1CR16CS093) _____

SUSHMITHA B.V (1CR16CS169) _____

ABSTRACT

Preserving the privacy of the digital biometric data, (e.g., face image) stored in a central database has become of paramount importance.

Biometric is commonly used in many automated verification systems offering several advantages over traditional methods .In today's world of multimedia applications, digital image security is of utmost importance. This work explores the possibility of using Elliptical cryptography and Arnold transformation. The aim of the project is to provide a secure system by implementing two levels of security (Arnold transformation and ECC).

Using Arnold transformation the is scrambled and fed to machine for learning. A biometric image is selected for identification using graphical user interface and secrambled using arnold transformation and identified as first level of authentication .When a person is identified a security pin asked as second level of transformation which is implemented using ECC.Arnold transformation is difficult to decode as it uses step by step scrambling, malicious user has to go through various steps to descramble and he goes wrong in any step he could not descramble the image.The second level of security that is ECC is very powerful cryptography method as compared to RSA.

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing me a platform to pursue my studies and carry out my final year project

I have great pleasure in expressing my deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore, for his constant encouragement.

I would like to thank **Dr. Prem Kumar Ramesh,** Professor and Head, Department of Computer Science and Engineering, CMRIT, Bangalore, who has been a constant support and encouragement throughout the course of this project.

I consider it a privilege and honor to express my sincere gratitude to my guide **Shivaraj V B, Assistant Professor,** Department of Computer Science and Engineering, for the valuable guidance throughout the tenure of this review.

I also extend my thanks to all the faculty of Computer Science and Engineering who directly or indirectly encouraged me.

Finally, I would like to thank my parents and friends for all their moral support they have given me during the completion of this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ECC	**Elliptic Curve Cryptography**
PCA	**Principal Complex Analysis**
ML	**Machine Learning**
VC	**Visual Cryptography**

CHAPTER 1

INTRODUCTION

Biometrics is a way to measure a person's physical characteristics to verify this identity. These can include physiological traits, such as fingerprints, voice, face, retina and iris patterns, hand geometry, gait or DNA profiles.

Biometric systems can seem complicated, but they all use the same three steps:

1 Enrolment: The first time you use a biometric system, it records basic information about you, like your name or an identification number. It then captures an image or recording of your specific trait.

2 Storage: Contrary to what you may see in movies, most systems don't store the complete image or recording. They instead analyse your trait and translate it into a code or graph. Some systems also record this data onto a smart card that you carry with you.

3 Comparison: The next time you use the system, it compares the trait you present to the information on file. Then, it either accepts or rejects that you are who you claim to be.

There are two parts to the employment of any biometric system. Firstly, biometric technologies capture and store characteristics in a database in order to identify an individual. Secondly, the information in this database is cross-referenced to verify or authenticate an individual's identity in a range of contexts.

Unlike the use of other forms of authentication, such as passwords or tokens, biometric recognition provides a strong link between an individual and a data record.

One area where biometrics can provide substantial help is in guarding against attempts to fraudulently establish multiple identities. By searching through the enrolled biometric references, individuals who appear to have previously enrolled using a different identity can be highlighted for further investigation. It is very difficult to perform this type of check without the use of biometrics.

1.1 Relevance of the Problem

- The purpose of most biometric systems is to authenticate a person's claimed identity authentication.
- Biometrics tend to be more convenient than other methods of identity authentication.
- Conventional Arnold scrambling has two levels of scrambling.
- Arnold scrambling for pixels and blocks of the plain image is used in our project.

1.2 Problem Statement

- To share information without revealing the message.
- In order to differ from conventional methods to hide data in the image, an algorithm can be developed using scrambling.
- Using Arnold Transformation and ECC provides high security for the image that is being transmitted.

1.3 Objectives

- The objective is to protect the Biometric and preserve it from the malicious users by considering Arnold scrambling and ECC.
- This proposes a two level image scrambling to increase the robustness of Arnold transform.
- Security analysis is carried out which shows that the encryption is highly secure. The original cover image is recoverable with minimal losses.

1.4 Scope of the Project

- To Train the Machine with multiple scrambled images of different people.
- To identify the scrambled image of the person as the first level of authentication.
- To match the pin entered by the identified person as the second level of authentication.

1.5 Methodology

- The idea behind our methodology is somewhat simple ,an input is asked from user to select his or her image
- The selected image is scrambled using the Arnold transformation as the ML model identifies only scrambled images.
- Using PCA is used to match the pattern of the scrambled image.
- The selected image subject's ID is recognised by the Machine learning model using random forest classifier.
- If the subject is identified with a high percentage of confidence then ECC based verification with a pin for a particular person.
- The encrypted pin stored in the database is decrypted and verified with the input .If the password is verified the access is granted in the system.

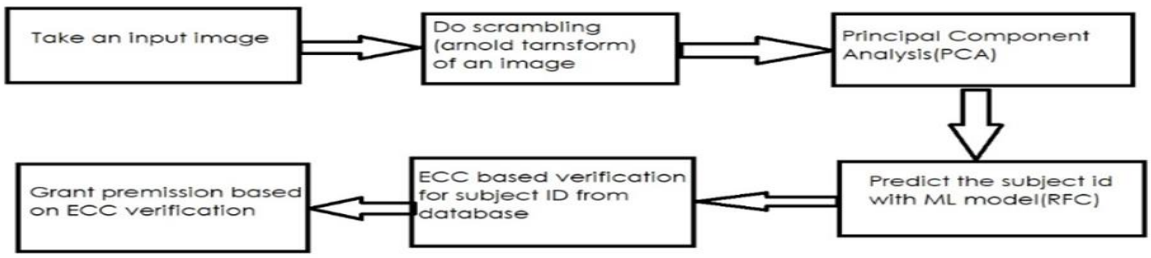


Figure 1.1: System architecture

Table 1.1 Summary of the Approaches of different VC's

Technique	Merit	Demerit
Traditional Visual Cryptography	**Provides security for binary images**	**Does not create meaningful image shares**
Extended Visual Cryptography	**Generates meaningful shares**	**Contrast loss occurs**
Progressive Visual Cryptography	**No pixel expansion**	**No guarantee on the correct reconstructio n**

Table 1.1 summarizes the comparisons of different VC's, their merits and demerits are also discussed.

CHAPTER 2

LITERATURE SURVEY

2.1 Paper 1: Visual Cryptography Using Half toned images

Visual cryptography encrypts visual information. It works on informative images, encrypts it, to what appears as a series of randomised pixels and decrypts it using a simple XOR operation.

In this paper the author has used

Simple Block replacement technique:

Simple block replacement or SBR follows a block wise concept. It first divides the image as shown in the figure into 2x2 blocks each containing 4 pixels working one block at a time SPR pre-processors image before the main process of producing image shares is initiated.

The algorithm is as follows:

1. Divide the image into a set of 2x2 blocks
2. Block Count
   Count the number of black pixels in the 2x2 block
3. IF Count >= 2
   Convert all pixels in the block to black
   ELSE
   Convert all pixels in the block to white
4. Perform steps 2 and 3 iteratively for all blocks within the image.

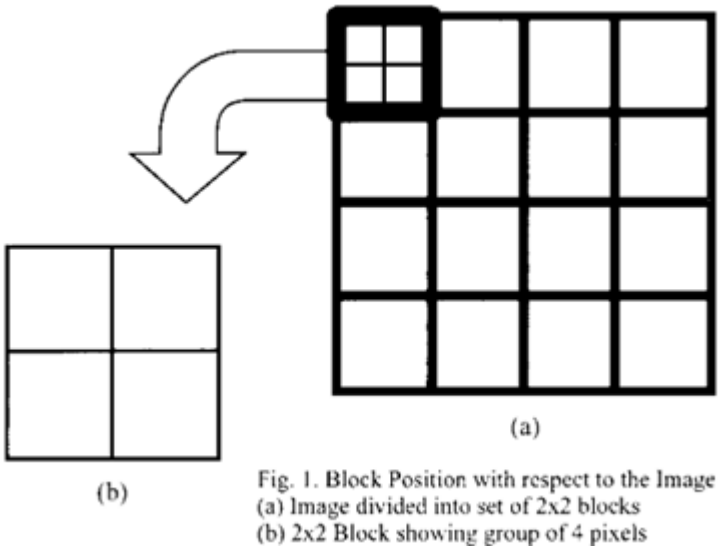


Fig. 1. Block Position with respect to the Image
(a) Image divided into set of 2x2 blocks
(b) 2x2 Block showing group of 4 pixels

Figure 2.1: Simple Block Replacement

Through this method the author can conclude that splitting the images into more than three shares includes the use of third or higher degree polynomial which on researching continuous to mention pixel expansion as a disadvantage using block replacement in these cases isn't feasible as would have to increase the block size from 2x2 to 3x3 and higher. Since blocks behave as the basic unit of image, increasing the block size will only make it more distorted with sharper edges and pixelated effects.

2.2 Paper 2:-Arnold Transform Based Image Scrambling Method

Arnold transformation is scrambling the location of pixels or blocks of the image .Its encryption algorithm is chaotic, so attackers cannot decipher it.

In this paper author has used

Improved Arnold Algorithm Model:-

Select a non-square $L \times L$ pixels area in a digital image, doing the Arnold transformation, in order to achieve parts of the region scrambling. Fig. 2 shows the idea of using improved Arnold scrambling algorithm for image scrambling in the single region:

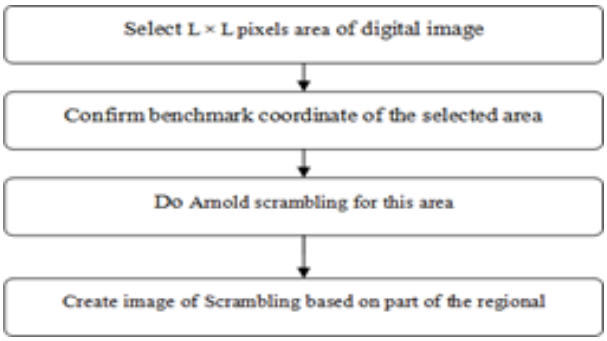


Fig. 2.2: Flow of digital image scrambling

2.3 Paper 3:- An Enhanced Elliptic Curve Cryptography for Biometric

In this paper, we study the usage of Elliptic Curves in cryptography. In biometric based system, persons should firstly register in the system. A procedure in which an input device gathers their biometric data and a master template is formed from this data and saved. At this point, in every identification case, the biometric data is gathered from the person and then a new template will be formed. After that this template will be compared against the master template and using a matching rate threshold the system makes a decision to allow or refuse the received identity.

A biometric signature is created via methods of producing a private key (secret key) from a biometric and utilizing this private key to form a digital signature.

The elliptic curve digital signature algorithm is divided into three main stages as follows:

Stage 1: Key Generation

Stage 2: Generation of Signature

Stage 3: Verification

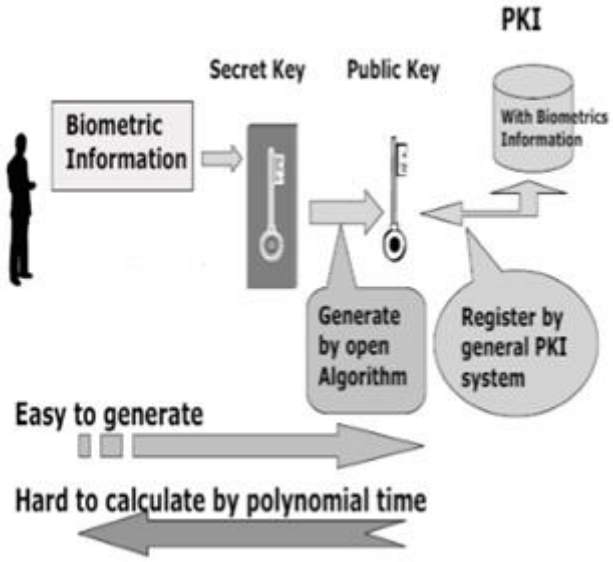


Figure 2.3:Elliptic curve digital signature

The main benefit of combining ECC and biometric is that there is no need to save biometric templates or private keys anywhere that can solve PKI's key management problem. The other benefit is that the ECDSA keys are smaller than RSA keys and with equal security that can enhance communication performance.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

This Chapter describes the requirements. It specifies software requirements that are in order to run the system properly. The Software Requirement Specification is explained in detail, which includes the overview of functional and non-functional requirements.

3.1 Functional Requirements

Functional Requirements defines the function of the system and gives the outline of what the system does.

- Ask for the image before granting access to the system and scramble the selected image for security purposes.
- Scrambled image is stored in a bmp file for the purpose of matching and identification from a stored dataset.
- Demand for pin or password for the purpose of a second level of authentication which uses ECC after the 1st level of scrambling authentication message pops up.

3.2 Non-Functional Requirements

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviours.

- The message pops up in the first level of authentication with the percentage of matching the selected image with the already stored ounces. If image does not match the access is denied.
- Pin is asked at second level after verification of first level is pin is not correct the access is denied with pop up message of denial.
- Every time when scrambling is done the selected image is stored in bmp file from where it is matched with the other stored image.

- Dataset of 15 people is stored as enrolled images and a dataset of those 15 images each with 11 expressions is used as an image to be selected.

3.3 Software Requirements

3.3.1 Language Used

Python

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system with reference counting.

Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release." No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.5.x and later are supported.

3.3.2 Packages used

- numpy
- pickle
- OpenCv
- re
- pillow
- TKinter
- sklearn
- matplotlib

Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

Syntax:

import numpy

Array creation

```
>>> import numpy as np

>>> x = np.array([1, 2, 3])

>>> x

array([1, 2, 3])

>>> y = np.arange(10)  # like Python's range, but returns an array

>>> y

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Basic operations

```
>>> a = np.array([1, 2, 3, 6])

>>> b = np.linspace(0, 2, 4)  # create an array with four equally spaced points starting with 0 and ending with 2.

>>> c = a - b

>>> c

array([ 1.        ,  1.33333333,  1.66666667,  4.        ])

>>> a**2

array([ 1,  4,  9, 36])
```

Pickle

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it "serializes" the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

Syntax:

```
import pickle
```

Advantages of using Pickle Module:

- Recursive objects (objects containing references to themselves): Pickle keeps track of the objects it has already serialized, so later references to the same object won't be serialized again. (The marshal module breaks for this.)
- Object sharing (references to the same object in different places): This is similar to self-referencing objects; pickle stores the object once, and ensures that all other references point to the master copy. Shared objects remain shared, which can be very important for mutable objects.

- User-defined classes and their instances: Marshal does not support these at all, but pickle can save and restore class instances transparently. The class definition must be importable and live in the same module as when the object was stored.

OpenCv

OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license.

OpenCV's application areas include:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking
- Augmented reality

Syntax:

import cv2

Regular expression (RE)

A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string, which comes down to the same thing).

Regular expressions can be concatenated to form new regular expressions; if *A* and *B* are both regular expressions, then *AB* is also a regular expression. In general, if a string *p* matches *A* and another string *q* matches *B*, the string *pq* will match AB. This holds unless *A* or *B* contain low precedence operations; boundary conditions between *A* and *B*; or have numbered group references. Thus, complex expressions can easily be constructed from simpler primitive expressions like the ones described here.

Regular expressions can contain both special and ordinary characters. Most ordinary characters, like 'A', 'a', or '0', are the simplest regular expressions; they simply match themselves. You can concatenate ordinary characters, so last matches the string 'last'. (In the rest of this section, we'll write RE's in this special style, usually without quotes, and strings to be matched 'in single quotes'.)

Some characters, like '|' or '(', are special. Special characters either stand for classes of ordinary characters, or affect how the regular expressions around them are interpreted.

Repetition qualifiers (*, +, ?, {m,n}, etc) cannot be directly nested. This avoids ambiguity with the non-greedy modifier suffix ?, and with other modifiers in other implementations. To apply a second repetition to an inner repetition, parentheses may be used. For example, the expression (?:a{6})* matches any multiple of six 'a' characters.

Syntax:

```
import re
```

Pillow

Python Imaging Library Pillow is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X .

Development appears to be discontinued, with the last commit to the PIL repository coming in 2011.Consequently, a successor project called Pillow has forked the PIL repository and added Python 3.x support. This fork has been adopted as a replacement for the original PIL in Linux distributions including Debian and Ubuntu

Pillow offers several standard procedures for image manipulation. These include:

- per-pixel manipulations,
- masking and transparency handling,
- image filtering, such as blurring, contouring, smoothing, or edge finding,
- image enhancing, such as sharpening, adjusting brightness, contrast or color,
- adding text to images and much more.

File format

Some of the file formats supported are- PPM, PNG, JPEG, GIF, TIFF, and BMP. It is also possible to create new file decoders to expand the library of file formats accessible

Syntax:

```
from PIL import Image, ImageFilter  # imports the library
original = Image.open("file.ppm") # load an image from the hard drive
 blurred = original.filter(ImageFilter.BLUR) # blur the image
 original.show() # display both images
blurred.show()
```

TKinter

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit,[1] and is Python's *de facto* standard GUI.Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python.

Some definitions

- Window

This term has different meanings in different contexts, but in general it refers to a rectangular area somewhere on the user's display screen.

- Top Level Window

A window that exists independently on the screen. It will be decorated with the standard frame and controls for the desktop manager. It can be moved around the desktop, and can usually be resized.

- Widget

The generic term for any of the building blocks that make up an application in a graphical user interface.

  - Core widgets: The containers: frame, labelframe, toplevel, paned window. The buttons: button, radiobutton, checkbutton (checkbox), and menubutton. The text widgets: label, message, text. The entry widgets: scale, scrollbar, listbox, slider, spinbox, entry (singleline), optionmenu, text (multiline), and canvas (vector and pixel graphics).
  - Tkinter provides three modules that allow pop-up dialogs to be displayed: tk.messagebox (confirmation, information, warning and error dialogs), tk.filedialog (single file, multiple file and directory selection dialogs) and tk.colorchooser (colour picker).
  - Python 2.7 and Python 3.1 incorporate the "themed Tk" ("ttk") functionality of Tk 8.5.This allows Tk widgets to be easily themed to look like the native desktop environment in which the application is running, thereby addressing a long-standing criticism of Tk (and hence of Tkinter). Some widgets are exclusive to ttk, such as the combobox, progressbar and treeview widgets.

Frame

In Tkinter, the Frame widget is the basic unit of organization for complex layouts. A frame is a rectangular area that can contain other widgets.

Child and parent

When any widget is created, a parent-child relationship is created. For example, if you place a text label inside a frame, the frame is the parent of the label.

Syntax

```
from tkinter import *
root = Tk()                                  # Create the root (base) window
w = Label(root, text="Hello, world!")        # Create a label with words
w.pack()                                     # Put the label into the window root.mainloop()
                                             # Start the event loop
```

SkLearn

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-learn is largely written in Python, and uses numpy extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in Cython to improve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR. In such cases, extending these methods with Python may not be possible.

Scikit-learn integrates well with many other Python libraries, such as matplotlib and plotly for plotting, numpy for array vectorization, pandas dataframes, scipy, and many more.

Syntax

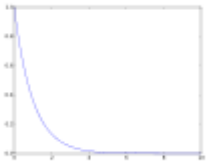
```
from sklearn.decomposition import PCA
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

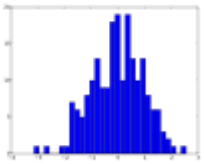
MatPlotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

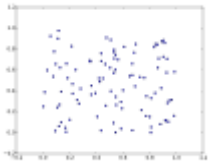
Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python 3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement.



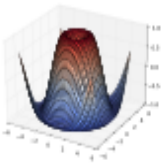
Line plot



Histogram



Scatter plot



3D plot

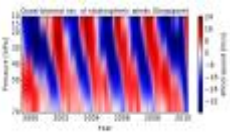
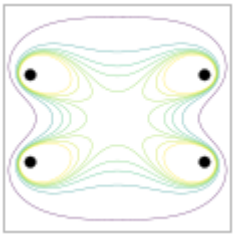
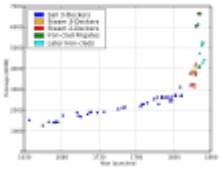


Image plot

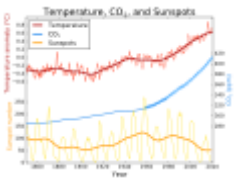


Contour plot

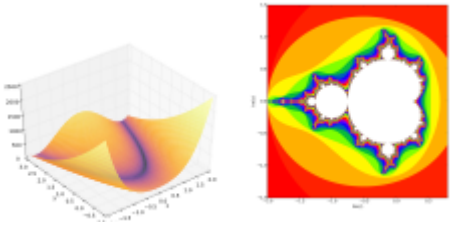


Scatter plot

Polar plot



Line plot



3-D plot **Image plot**

Figure 3.1:Example of Matlib

3.3.3 Machine learning Methods

- Principal Complex Analysis
- Random forest Classifier

Principal Complex Analysis

Principal Component Analyis is basically a statistical procedure to convert a set of observation of possibly correlated variables into a set of values of linearly uncorrelated variables.

Each of the principal components is chosen in such a way so that it would describe most of the still available variance and all these principal components are orthogonal to each other. In all principal components first principal component has maximum variance.

Uses of PCA:

- It is used to find inter-relation between variables in the data.
- It is used to interpret and visualize data.
- As number of variables are decreasing it makes further analysis simpler.
- It's often used to visualize genetic distance and relatedness between populations.

These are basically performed on square symmetric matrix. It can be a pure sums of squares and cross products matrix or Covariance matrix or Correlation matrix. A correlation matrix is used if the individual variance differs much.

Principal Axis Method: PCA basically search a linear combination of variables so that we can extract maximum variance from the variables. Once this process completes it removes it and search for another linear combination which gives an explanation about the maximum proportion of remaining variance which basically leads to orthogonal factors. In this method, we analyze total variance.

Eigenvector: It is a non-zero vector that stays parallel after matrix multiplication. Let's suppose x is eigen vector of dimension r of matrix M with dimension r*r if Mx and x are parallel. Then we need to solve Mx=Ax where both x and A are unknown to get eigen vector and eigen values.

Under Eigen-Vectors we can say that Principal components show both common and unique variance of the variable. Basically, it is variance focused approach seeking to reproduce total variance and correlation with all components. The principal components are basically the linear combinations of the original variables weighted by their contribution to explain the variance in a particular orthogonal dimension.

Eigen Values: It is basically known as characteristic roots. It basically measures the variance in all variables which is accounted for by that factor. The ratio of eigenvalues is the ratio of explanatory importance of the factors with respect to the variables. If the factor is low then it is contributing less in explanation of variables. In simple words, it measures the amount of variance in the total given database accounted by the factor. We can calculate the factor's eigen value as the sum of its squared factor loading for all the variables.

Now, Let's understand Principal Component Analysis with Python.

To get the dataset used in the implementation

Step 1: Importing the libraries

```
>>>import numpy as np

>>>import matplotlib.pyplot as plt

>>>import pandas as pd
```

Step 2: Importing the data set

Import the dataset and distributing the dataset into X and y components for data analysis.

```
# importing or loading the dataset

dataset = pd.read_csv('wines.csv')

# distributing the dataset into two components X and Y

X = dataset.iloc[:, 0:13].values

y = dataset.iloc[:, 13].values
```

Step 3: Splitting the dataset into the Training set and Test set

```
# Splitting the X and Y into the

# Training set and Testing set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Step 4: Feature Scaling

Doing the pre-processing part on training and testing set such as fitting the Standard scale.

```
filter_none

brightness_4

# performing preprocessing part

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()


X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)
```

Step 5: Applying PCA function

Applying the PCA function into training and testing set for analysis.

filter_none

brightness_4

```
# Applying PCA function on training
# and testing set of X component
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
```

Step 6: Fitting Logistic Regression To the training set

filter_none

brightness_4

```
# Fitting Logistic Regression To the training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

Step 7: Predicting the test set result

filter_none

brightness_4

```
# Predicting the test set result using
# predict function under LogisticRegression
y_pred = classifier.predict(X_test)
```

Step 8: Making the confusion matrix

filter_none

brightness_4

```
# making confusion matrix between
#  test set of Y and predicted value.
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Step 9: Predicting the training set result

filter_none

brightness_4

```
# Predicting the training set
# result through scatter plot
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                     stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
                     stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
            X2.ravel()]).T).reshape(X1.shape), alpha = 0.75,
            cmap = ListedColormap(('yellow', 'white', 'aquamarine')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
```

```
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green', 'blue'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('PC1') # for Xlabel
plt.ylabel('PC2') # for Ylabel
plt.legend() # to show legend
# show scatter plot
plt.show()
```

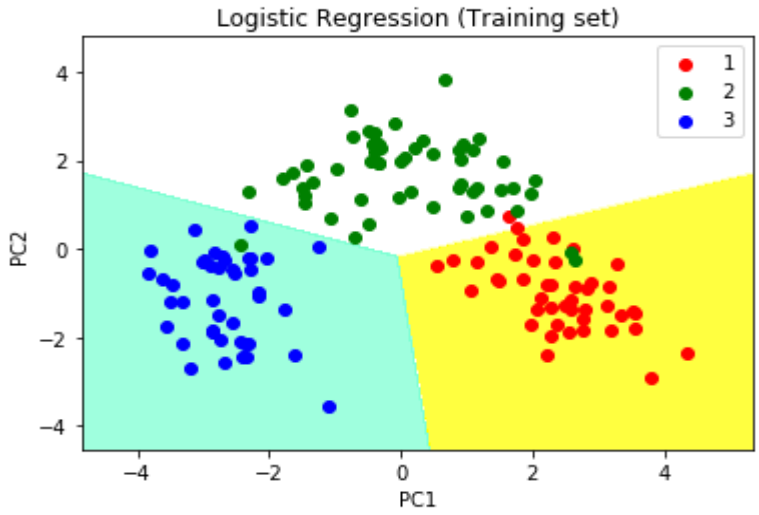


Figure 3.2: Logistic regression

Step 10: Visualising the Test set results

filter_none

brightness_4

```
# Visualising the Test set results through scatter plot
from matplotlib.colors import ListedColormap
```

```
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                     stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
                     stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
             X2.ravel()]).T).reshape(X1.shape), alpha = 0.75,
             cmap = ListedColormap(('yellow', 'white', 'aquamarine')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green', 'blue'))(i), label = j)
# title for scatter plot
plt.title('Logistic Regression (Test set)')
plt.xlabel('PC1') # for Xlabel
plt.ylabel('PC2') # for Ylabel
plt.legend()
# show scatter plot
plt.show()
```

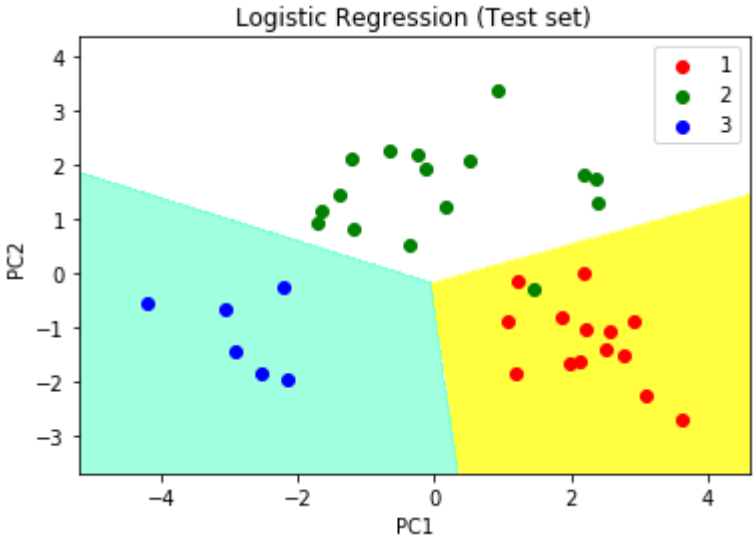


Figure 3.3: Logistic regression

Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

We can understand the working of Random Forest algorithm with the help of following steps –

- Step 1 − First, start with the selection of random samples from a given dataset.
- Step 2 − Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- Step 3 − In this step, voting will be performed for every predicted result.

- Step 4 – At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working –

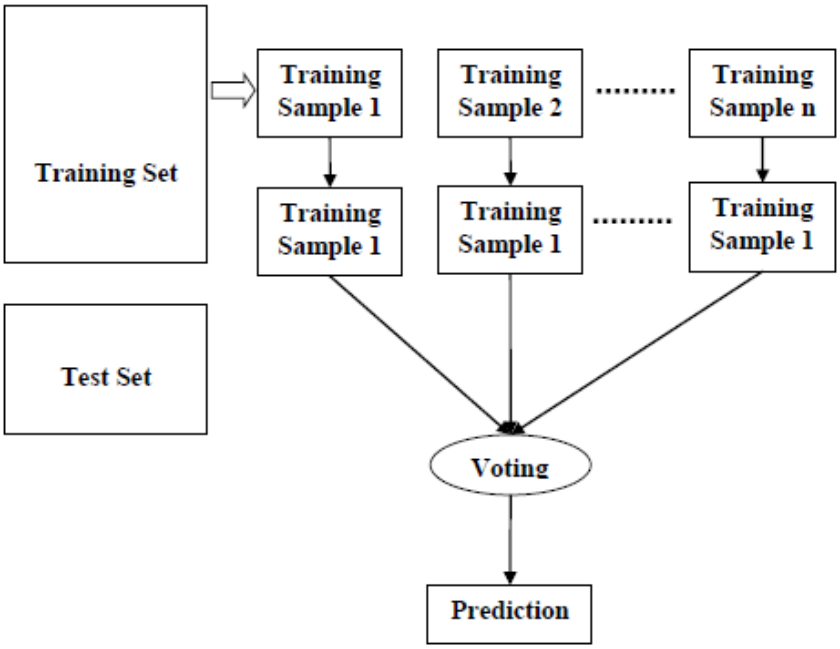


Figure 3.4: Working of Random Forest Classifier

3.4 Hardware Requirements

• RAM-8GB

• Processor-Intel

• Hard Disk-1TB

• Keyboard

• Monitor

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

System Analysis is the process that decomposes a system into its component pieces for the purpose of defining how well those components interact to accomplish the set requirements. Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. System design is the process of defining and developing systems to satisfy specified requirements of the user.The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

Elements of a System

• Architecture - This is the conceptual model that defines the structure, behavior and more views of a system. We can use flowcharts to represent and illustrate the architecture.

• Modules - This are components that handle one specific tasks in a system. A combination of the modules make up the system.

• Components - This provides a particular function or group of related functions. They are made up of modules.

• Interfaces - This is the shared boundary across which the components of a the system exchange information and relate.

• Data - This the management of the information and data flow.

4.1 System Architecture

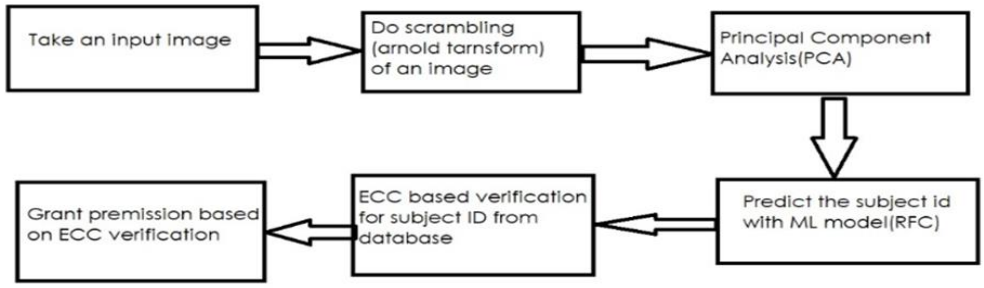


Figure 4.1: System architecture

Step 1: An input is asked from user to select his or her image

Step 2: The selected image is scrambled using Arnold transformation as ML model identifies only scrambled images.

Step 3: Using PCA is used to match the pattern of the scrambled image.

Step 4: The selected image subject's ID is recognised by the Machine learning model using random forest classifier.

Step 5: If the subject is identified with a high percentage of confidence then ECC based verification with a pin for a particular person.

Step 6: The encrypted pin stored in the database is decrypted and verified with the input .If the password is verified the access is granted in the system.

4.2 Dataset

A data set (or dataset) is a collection of data. In the case of tabular data, a data set corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the data set in question. The data set lists values for each

of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a datum. Data sets can also consist of a collection of documents or files. YALEFACES are databases of numerical synthetic images easily usable to evaluate in a systematic way the performance of visibility and contrast restoration algorithms. YALEFACES dataset in which the 15 people with 11 different expressions for each person.

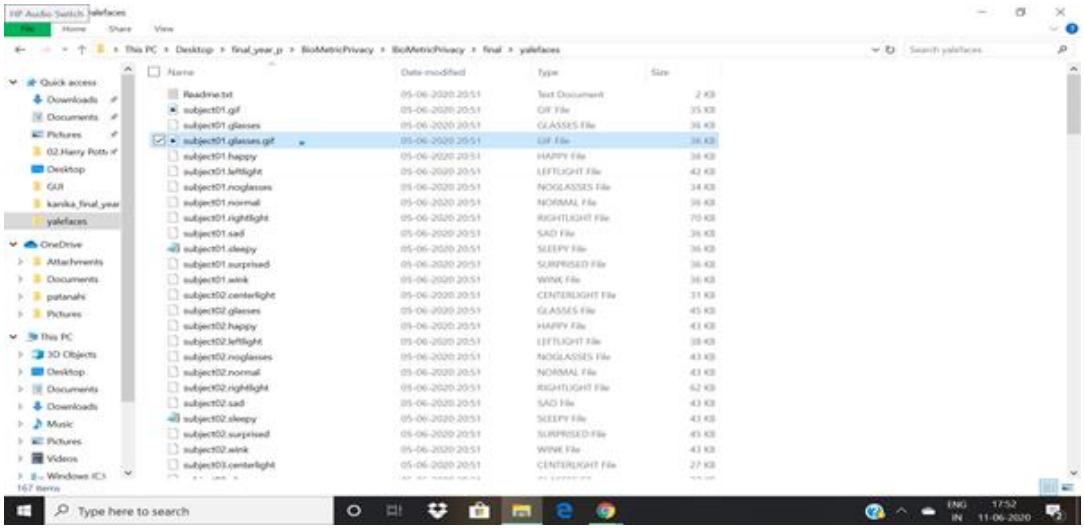


Figure 4.2 Screenshot of Dataset

4.3 Data Flow of Use Case

Dataflow is a fully managed streaming analytics service that minimizes latency, processing time, and cost through auto scaling and batch processing.

• In order to verify the effectiveness of our proposed methodology we are applying Random forest classifier feature extraction on YALEFACES synthetic database and it is used for extracting scrambled images.

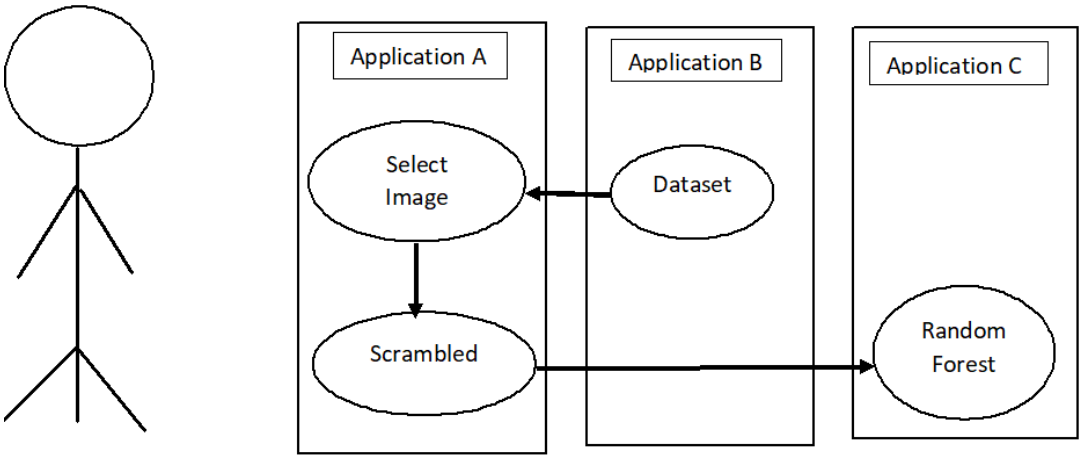


Figure 4.3: Dataflow of use case Authentication 1

- The image is selected from dataset by user for authentication
- The selected image is scrambled by the user .
- The scrambled image is fed to a machine for identification.
- The fed image is identified using random forest.
- The result of the prediction is sent to the user.

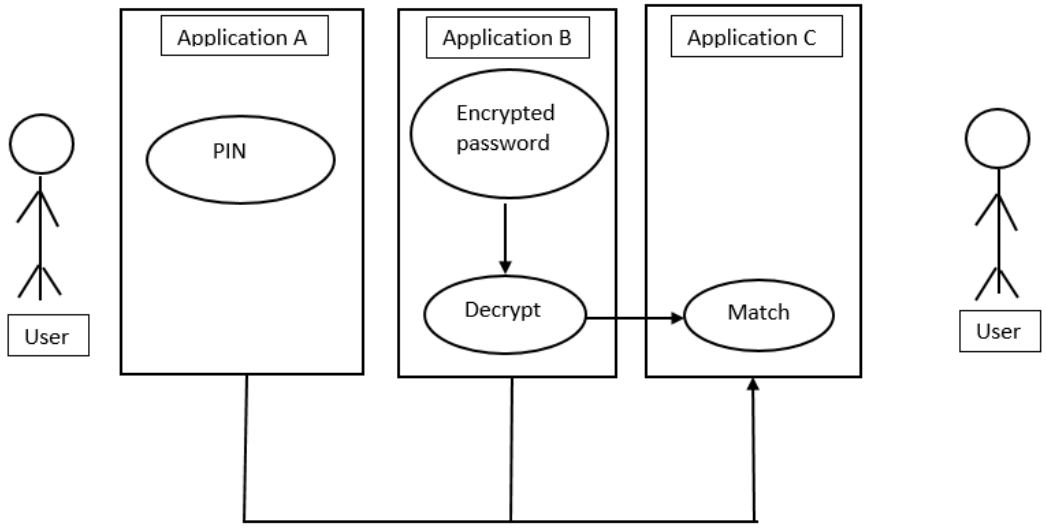


Figure 4.4: Dataflow of use case Authentication 2

- This happens when first authentication is successful .

- The identified user is asked for password as second level of authentication.
- The user entry is sent to the system where password is encrypted.
- The stored encrypted password is decrypted and matched with the user's entry.
- If it matches the access is granted and if it doesn't , access is denied .

CHAPTER 5

IMPLEMENTATION

5.1 Arnold Transform:

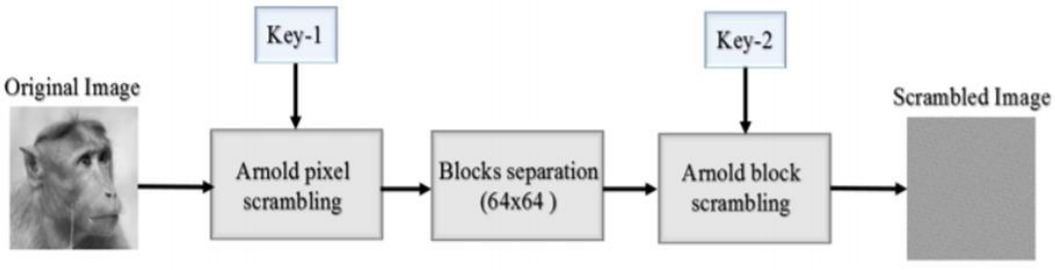


Figure 5.1: Scrambling Process

Arnold scrambling for pixels and blocks. In the first level, Arnold pixel scrambling is performed on the original image of size N × N, where the pixels are scrambled based on the number of iterations. In the second level the scrambled image is divided into 8×8 blocks for which Arnold scrambling is conducted block-wise.

Arnold scrambling transforms the position of a pixel from (x, y) to a new position (s, t) . The position of pixels changes from one point to another based on the equation

[x y] = [1 1 1 2][s t](mod 1)

In order to apply the transformation to a digital image the term mod 1 can be replaced by mod N where N is the size of the digital image.

[x y] = [1 1 1 2][s t] (mod N)

The transformation matrix of Arnold scrambling is a mod2 transformation matrix which is given by A. (x, y) T is the input and (s, t) T is the output.

5.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality-reduction technique that is often used to transform a high-dimensional dataset into a smaller-dimensional subspace prior to running a machine learning algorithm on the data.

It is often helpful to use a dimensionality-reduction technique such as PCA prior to performing machine learning because:

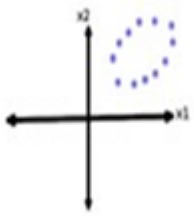
1. Reducing the dimensionality of the dataset reduces the size of the space on which k-nearest-neighbors (k\KNN) must calculate distance, which improve the performance of KNN.

2. Reducing the dimensionality of the dataset reduces the number of degrees of freedom of the hypothesis, which reduces the risk of over fitting.

3. Most algorithms will run significantly faster if they have fewer dimensions they need to look at.

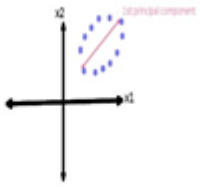
4. Reducing the dimensionality via PCA can simplify the dataset, facilitating description, visualization, and insight.

Principal Component Analysis does just what it advertises; it finds the principal components of the dataset. PCA transforms the data into a new, lower-dimensional subspace—into a new coordinate system—. In the new coordinate system, the first axis corresponds to the first principal component, which is the component that explains the greatest amount of the variance in the data.



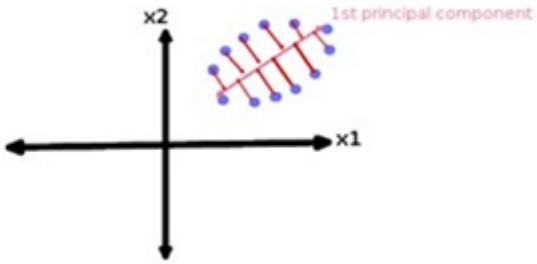
1. Let’s say your original dataset has two variables, x1 and x2:

2. Now, we want to identify the first principal component that has explained the highest amount of variance. Graphically, if we draw a line that splits the oval lengthwise, that line signifies the component that explains the most variance:

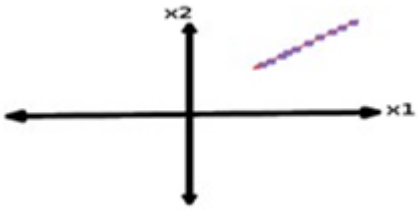


3. Let's say we just wanted to project the data onto the first principal component only. In other words, we wanted to use PCA to reduce our two-dimensional dataset onto a one dimensional dataset.

4. Basically, we would collapse our dataset onto a single line (by projecting it onto that line). The single line is the first principal component. Here is a picture:



5. Here is what the dataset will look like after being projected onto a single dimension corresponding to the first principal component:



6. We can see that we have destroyed some of the original information when we went from a two-dimensional dataset to the one-dimensional projection.

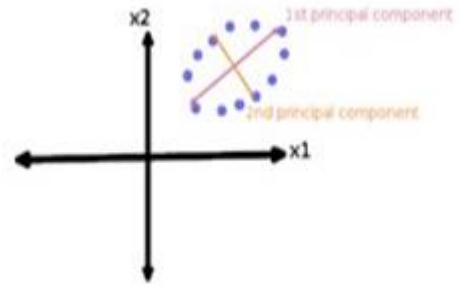
7. You can think of this sort of like a shadow.

8. Although we lost some information in the transformation, we did keep the most important axis, which incorporates information from both x1 and x2.

Second principal component

1. The second principal component must be orthogonal to the first principal component. In other words, it does its best to capture the variance in the data that is not captured by the first principal component. For our two-dimensional dataset, there can be only two principal components.

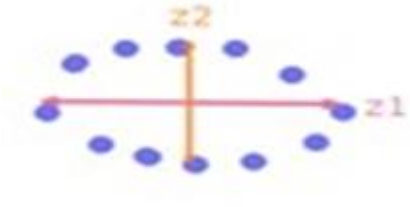
2. Here is a picture of the data and its first and second principal components:



3. Again, you can see that the two principal components are perpendicular to each other. They capture independent elements of the dataset.

4. If we were to perform PCA on this dataset and project the original dataset onto the first two principal components, then no information would be lost. (We are transforming from a two-dimensional dataset to a new two-dimensional dataset.) Instead, we would be merely rotating the data to use new dimensions.

5. Here is a picture of what the dataset looks like projected onto the first two principal components:



In general, the data will tend to follow the 80/20 rule. Most of the variance (interesting part of data) will be explained by a very small number of principal components. You might be able to explain 95% of the variance in your dataset using only 10% of the original number of attributes. However, this is entirely dependent on the dataset. Often, a good rule of thumb is to identify the principal components that explain 99% of the variance in the data. You cannot have more principal components than the number of attributes in the original dataset.

5.3 Elliptic curve Cryptography (ECC)

Elliptic curve cryptography, or ECC, is a powerful approach to cryptography and an alternative method from the well known RSA. It is an approach used for public key encryption by utilizing the mathematics behind elliptic curves in order to generate security between key pairs. ECC has been slowly gaining in popularity over the past few years due to it's ability to provide the same level of security as RSA with a much smaller key size.

The resources available to crack encrypted keys continues to expand, meaning the size of encrypted keys must continue to grow in order to remain secure. This can prove to be a burden to certain devices, particularly mobile, that do not have as much available computational power. However, elliptic curve cryptography helps to solve that problem.

An elliptical curve can simply illustrated as a set of points defined by the following equation:

$$y^2 = x^3 + ax + b$$

Based on the values given to *a* and *b*, this will determine the shape of the curve. Elliptical curve cryptography uses these curves over finite fields to create a secret that only the private key holder is able to unlock. The larger the key size, the larger the curve, and the harder the problem is to solve.

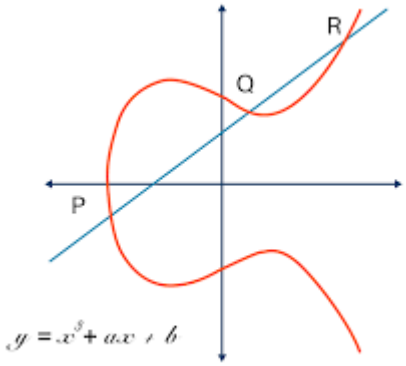


Figure 5.2: ECC curve

5.4 Random Forest Classifier

It is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees from a randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object.

Types of Random Forest models:

1. Random Forest Prediction for a classification problem:

f(x) = majority vote of all predicted classes over B trees

2. Random Forest Prediction for a regression problem:

f(x) = sum of all sub-tree predictions divided over B trees

Implementation of Random Forest Classification on real life dataset:

1. Importing Python Libraries and Loading our Dataset into a Data Frame

2. Splitting our dataset into training set and test set

3. Creating a Random Forest Regression model and fitting it to the training data

4. Predicting the test set results and making the Confusion matrix

5.5 Code

Importing all the useful packages

```
from tkinter import *
import tkinter
from PIL import Image
from PIL import ImageTk
from tkinter import filedialog
import ntpath
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pickle
import re
```

For classification of given subject

```
from arnoldClf import classf

import sys

loaded_image = [0]

predicted = 0

def button_12():
    txt = tb12.get()
    print(txt)
def button_14():
    txt = tb14.get()
    print(txt)

def eccVerify():
```

```
    global predicted
    intext = tb12.get()
    intext = str.encode(intext)
    from ecies.utils import generate_eth_key
    from ecies import encrypt, decrypt
    import binascii

    privKey = generate_eth_key()
    privKeyHex = privKey.to_hex()
    pubKeyHex = privKey.public_key.to_hex()
    print("Encryption public key:", pubKeyHex)
    print("Decryption private key:", privKeyHex)

    #plaintext = 'IamIn'
    passwordList = ["IamIn", "WhoIsThat", "NoWhere", "GoToHeaven", "NeverAgain",
"PiEqual", "NoNo", "Zero", "BigBig", "Jeroen"]
    plaintext = passwordList[predicted-1]
    print("predicted", predicted)

    plaintext = str.encode(plaintext)
    print("Plaintext:", plaintext)

    encrypted = encrypt(pubKeyHex, plaintext)
    print("Encrypted:", binascii.hexlify(encrypted))

    decrypted = decrypt(privKeyHex, encrypted)
    print("Decrypted:", decrypted)

    validate = (decrypted == intext)
    print(validate)
    if(validate == True):
```

```
        trans = 1
    if(validate == False):
        trans = 0
    trans_lab = ["Fail", "Success"]

    btn_14 = Button(app, text="Transaction: "+trans_lab[trans], command=button_14, height=2,
width=18)
    btn_14.grid(column=2, row=12)
```

Here, we are creating our class, Window, and inheriting from the Frame

```
# class. Frame is a class from the tkinter module. (see Lib/tkinter/__init__)
class Window(Frame):
    Define settings upon initialization. Here you can specify
    def __init__(self, master=None):
        # parameters that you want to send through the Frame class.
        Frame.__init__(self, master)

        #reference to the master widget, which is the tk window
        self.master = master

        #with that, we want to then run init_window, which doesn't yet exist
        self.init_window()

        self.w = 500
        self.h = 300

        self.load = Image.open("dummy.jpg")
        self.load = self.load.resize((self.w, self.h), Image.ANTIALIAS)
        render = ImageTk.PhotoImage(self.load)

        # labels can be text or images
```

```
        self.img = Label(root, image=render)
        self.img.place(x=0, y=0)
        self.text2 = Text(root, height=20, width=50)
        self.scroll = Scrollbar(root, command=self.text2.yview)
```

Creation of init_window

```
    def init_window(self):

        # changing the title of our master widget
        self.master.title("GUI")

        # allowing the widget to take the full space of the root window
        self.pack(fill=BOTH, expand=1)
```

Creating a menu instance

```
        menu = Menu(self.master)
        self.master.config(menu=menu)

        # create the file object)
        file = Menu(menu)

        # adds a command to the menu option, calling it exit, and the
        # command it runs on event is client_exit

        file.add_command(label="Load Test Image",command=self.showRealTimeImage)
        file.add_command(label="Arnold Transform",command=self.showArnoldTransf)
        file.add_command(label="Show Results", command=self.showRes)
        file.add_command(label="Exit", command=self.client_exit)

        #added "file" to our menu
        menu.add_cascade(label="File", menu=file)
    def showRealTimeImage(self):
```

```
        self.text2.pack_forget()

        filename1 = filedialog.askopenfilename()
        loaded_image[0] = filename1

        load = Image.open(filename1)
        resized_load = load.resize((self.w, self.h), Image.ANTIALIAS)
        render = ImageTk.PhotoImage(resized_load)

        # labels can be text or images
        self.img.configure(image=render)
        self.img.image = render
        self.img.place(x=0, y=0)

    def ECCKey(self):
        secret.key()

        filename1 = filedialog.askopenfilename()
        loaded_image[0] = filename1

        load = Image.open(filename1)
        resized_load = load.resize((self.w, self.h), Image.ANTIALIAS)
        render = ImageTk.PhotoImage(resized_load)

        # labels can be text or images
        self.img.configure(image=render)
        self.img.image = render
        self.img.place(x=0, y=0)


    def arnoldTransform(self, imageFileName):
```

```
    # Read an input image
    im = np.array(Image.open(imageFileName))

    # Read the number of rows of image
    N = im.shape[0]

    # Create a NxN meshgrid
    x,y = np.meshgrid(range(N),range(N))

    Generate maps
    xmap = (2*x+y) % N
    ymap = (x+y) % N
    5 iterations to generate feature representations
    for i in range(1,10):
        scr_img = Image.fromarray(im)
        im = im[xmap,ymap]
    scr_img.save("arnold.bmp")

def showArnoldTransf(self):

    self.arnoldTransform(loaded_image[0])

    load = Image.open("arnold.bmp")
    load = load.resize((self.w, self.h), Image.ANTIALIAS)
    self.render = ImageTk.PhotoImage(load)

    # labels can be text or images
    self.img.configure(image=self.render)
    self.img.image = self.render
    self.img.place(x=0, y=0)
```

```
    def showRes(self):

        global predicted
        pred, confid = classf(loaded_image[0])

        load = Image.open(loaded_image[0])
        predicted = pred
        resized_load = load.resize((self.w//2, self.h), Image.ANTIALIAS)
        render = ImageTk.PhotoImage(resized_load)

        # labels can be text or images
        self.img.configure(image=render)
        self.img.image = render
        self.img.pack(side=LEFT)

        self.text2.delete(1.0,END)
        self.text2.configure(yscrollcommand=self.scroll.set)
        self.text2.tag_configure('bold_italics', font=('Arial', 12, 'bold', 'italic'))
        self.text2.tag_configure('big', font=('Verdana', 20, 'bold'))
        self.text2.tag_configure('color', foreground='#476042',
                                 font=('Tempus Sans ITC', 12, 'bold'))
        self.text2.insert(END,'\n'+ str(pred) +'\n\n', 'big')
        quote = "The subject was identified to be "+ str(pred) + " with confidence: " +
str(round(confid, 2))
        self.text2.insert(END, quote, 'color')
        self.text2.pack(side=LEFT)
        self.scroll.pack(side=RIGHT, fill=Y)
    def client_exit(self):
        sys.exit(0)
```

Root window Created

```
# root window created. Here, that would be the only window, but
root = Tk()  # A root window for displaying objects
#root.geometry("500x300")
root.geometry("500x500")

#creation of an instance
app = Window(master=root)

btn14 = Button(app, text='Verify', command=eccVerify, height=3, width=10)
btn14.grid(column=2, row=4)

btn12 = Button(app, text='Password', command=button_12)
btn12.grid(column=2, row=2)

tb12 = Entry(app)
tb12.grid(column=2, row=3)



app.mainloop()
root.destroy()
sys.exit(0)
```

CHAPTER 6

RESULTS AND DISCUSSION

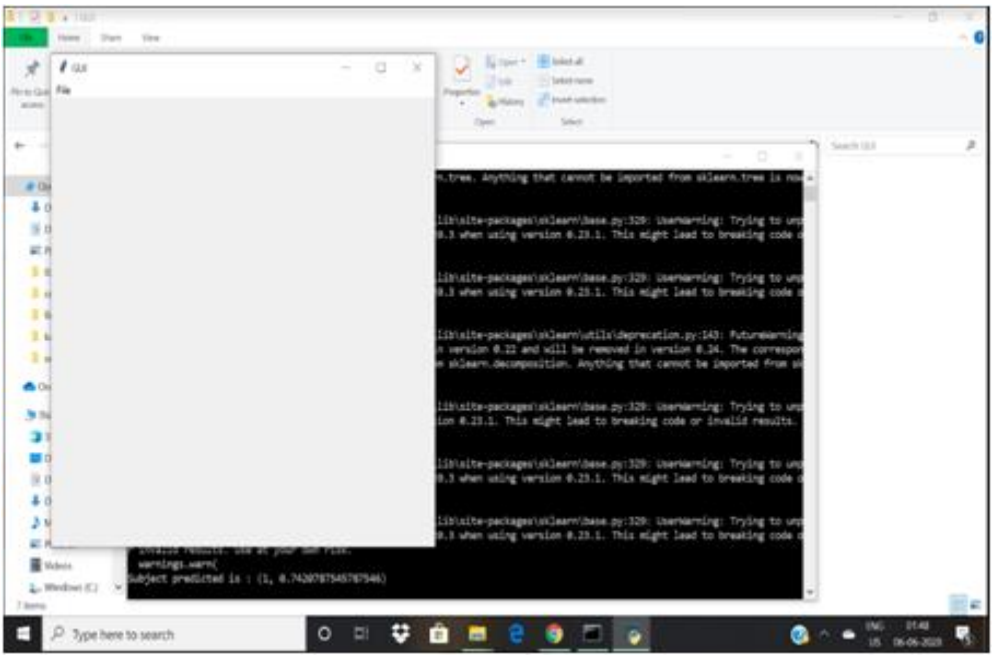


Fig 6.1: This is the home page

The above screenshot figure 6.1 shows the home page to select an image and to apply Arnold transformation.

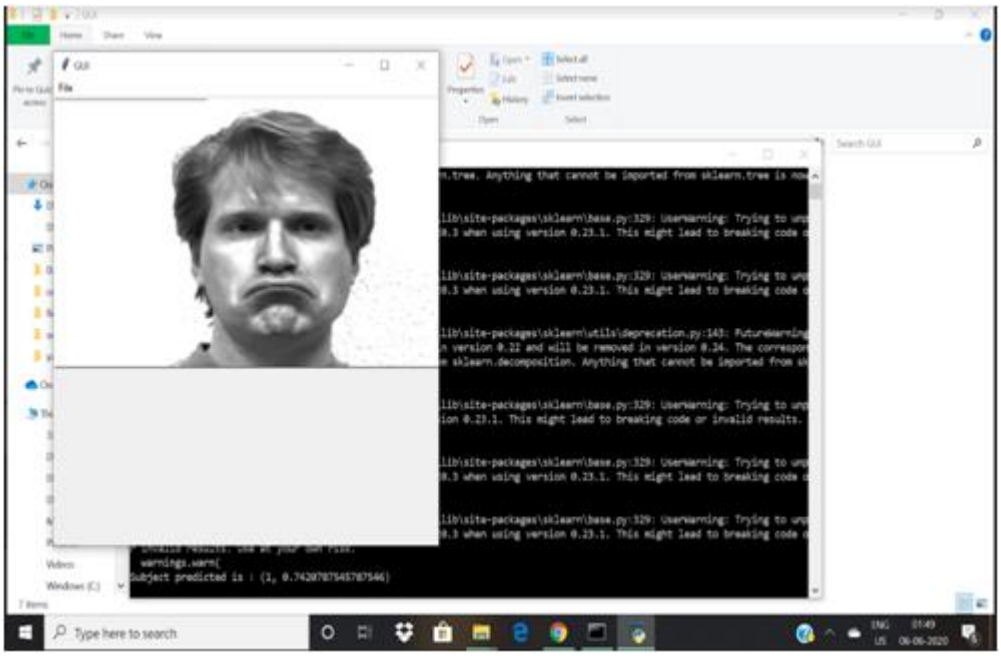


Fig 6.2: Image to be encrypted is selected

The above Fig 6.2 shows the image which is selected for identification and before that it needs to be scrambled using Arnold transformation.

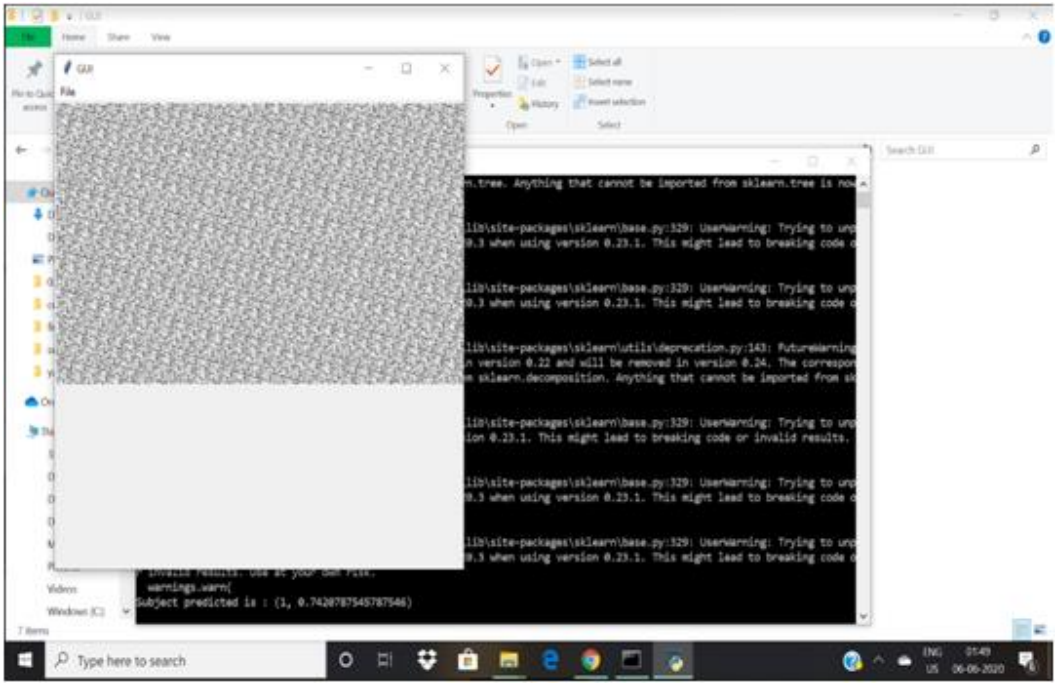


Figure 6.3: Image being encrypted using pixel and block scrambling techniques

The selected image which is shown in Fig 6.2 is scrambled using arnold transformation shown in Fig 6.3 .This image is now fed to the system for prediction using Random forest technique.

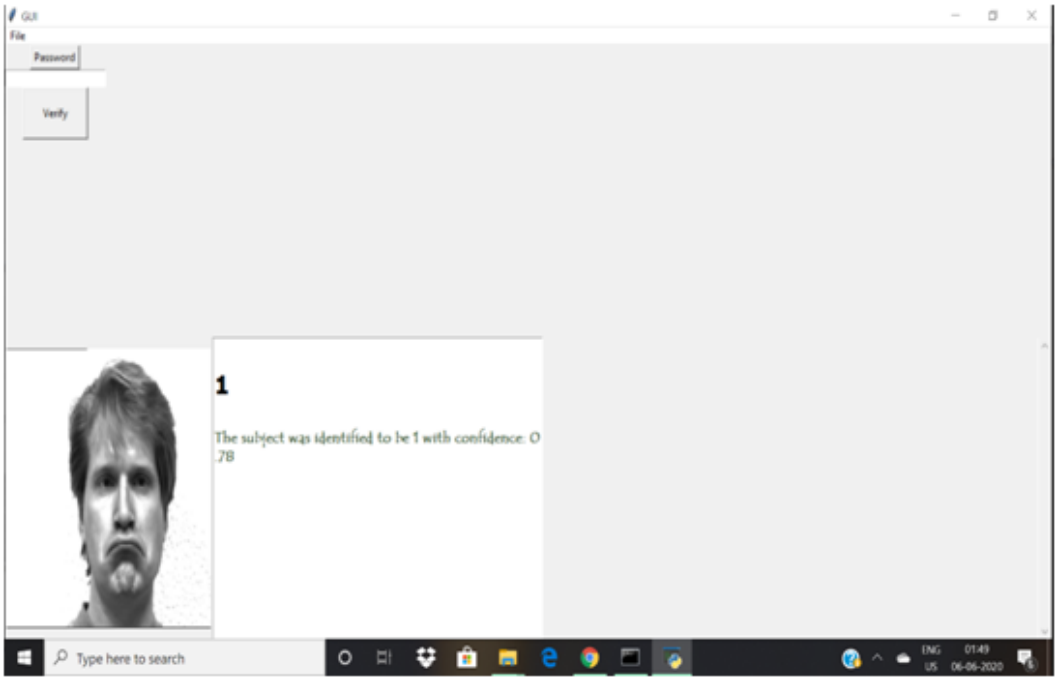


Figure 6.4: ECC password to verify the user

This figure 6.4 shows that the scrambled subject fig 6.3 is identified with confidence of 78%.This fig 6.4 is also asking for pin which is second level of implementation.

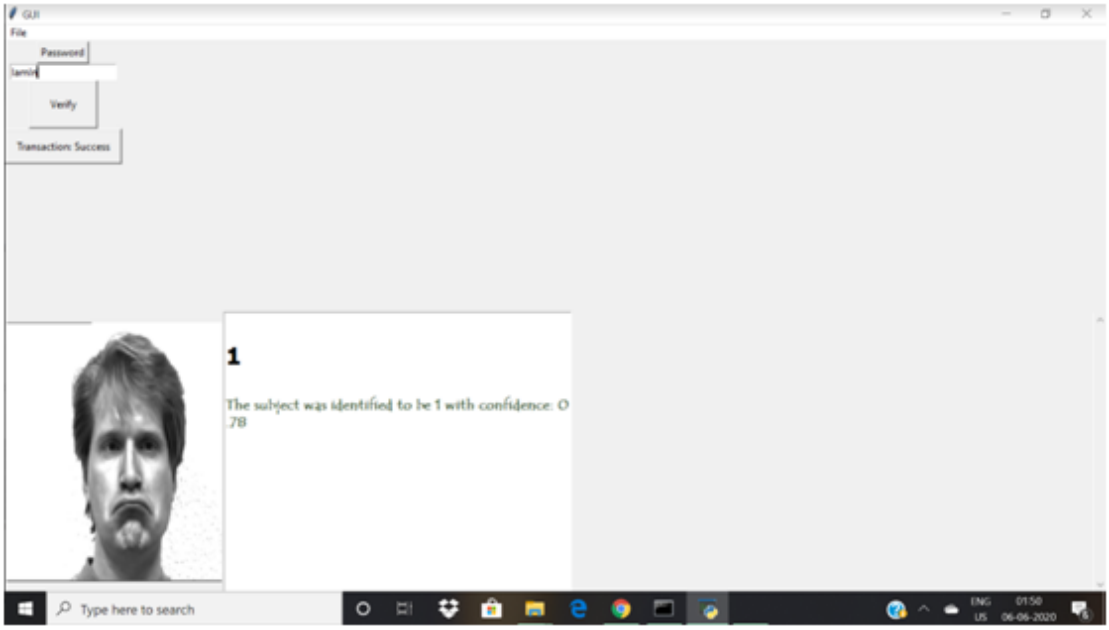


Figure 6.5: Password Successful

This fig 6.5 shows that password entered by user is correct hence access is granted.

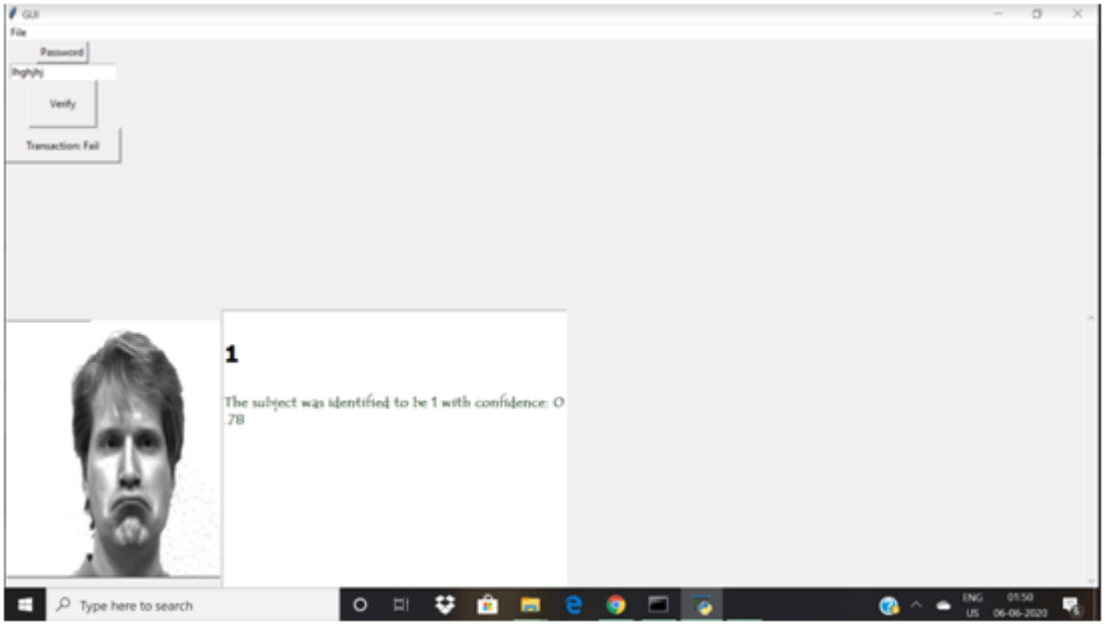


Figure 6.6: Password failed

This figure 6.6 shows that the password is not matched so access is denied.

CHAPTER 7

TESTING

Software testing defines an activity to check whether the actual results match the expected results and to ensure that the system is defect free.

7.1 Functional vs. Non-functional Testing

The goal of utilizing numerous testing methodologies in your development process is to make sure your software can successfully operate in multiple environments and across different platforms. These can typically be broken down between functional and nonfunctional testing. Functional testing involves testing the application against the business requirements. It incorporates all test types designed to guarantee each part of a piece of software behaves as expected by using uses cases provided by the design team or business analyst. These testing methods are usually conducted in order and include:

• Unit testing

• Integration testing

• System testing

• Acceptance testing

Non-functional testing methods incorporate all test types focused on the operational aspects of a piece of software.

These include:

• Performance testing

• Security testing

• Usability testing

• Compatibility testing

7.2 Unit Testing

Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test- driven environment will typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing can be conducted manually, but automating the process will speed up delivery cycles and expand test coverage. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process. Test left is a tool that allows advanced testers and developers to shift left with the fastest test automation tool embedded in any IDE.

7.3 Integration Testing

After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as a group through integration testing to ensure whole segments of an application behave as expected (i.e., the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and usually consist of a combination of automated functional and manual tests.

7.4 System Testing

System testing is a black box testing method used to evaluate the completed and integrated system, as a whole, to ensure it meets specified requirements. The functionality of the software is tested from end-to-end and is typically conducted by a separate testing team than the development team before the product is pushed into production.

7.5 Acceptance Testing

Acceptance testing is the last phase of functional testing and is used to assess whether or not the final piece of software is ready for delivery. It involves ensuring that the product is in compliance with all of the original business criteria and that it meets the end user's needs. This requires the product be tested both internally and externally, meaning you'll need to get it into the hands of your end users for beta testing along with those of your QA team. Beta testing is key to getting real feedback from potential customers and can address any final usability concerns.

7.6 Performance Testing

Performance testing is a non-functional testing technique used to determine how an application will behave under various conditions. The goal is to test its responsiveness and stability in real user situations. Performance testing can be broken down into four types:

• Load testing is the process of putting increasing amounts of simulated demand on your software, application, or website to verify whether or not it can handle what it's designed to handle.

• Stress testing takes this a step further and is used to gauge how your software will respond at or beyond its peak load. The goal of stress testing is to overload the application on purpose until it breaks by applying both realistic and unrealistic load scenarios. With stress testing, you'll be able to find the failure point of your piece of software.

• Endurance testing, also known as soak testing, is used to analyse the behaviour of an application under a specific amount of simulated load over longer amounts of time. The goal is to understand how your system will behave under sustained use, making it a longer process than load or stress testing (which are designed to end after a few hours). A critical piece of endurance testing is that it helps uncover memory leaks.

• Spike testing is a type of load test used to determine how your software will respond to substantially larger bursts of concurrent user or system activity over varying amounts of time. Ideally, this will help you understand what will happen when the load is suddenly and drastically increased.

7.7 Security Testing

1. With the rise of cloud-based testing platforms and cyber-attacks, there is a growing concern and need for the security of data being used and stored in software. Security testing is a non-functional software testing technique used to determine if the information and data in a system is protected. The goal is to purposefully find loopholes and security risks in the system that could result in unauthorized access to or the loss of information by probing the application for weaknesses. There are multiple types of this testing method, each of which aimed at verifying.

2. Confidentiality

3. Authentication

4. Authorization

5. Availability

6. Non-repudiation x basic principles of security

7.8 Usability Testing

Usability testing is a testing method that measures an application's ease-of-use from the end- user perspective and is often performed during the system or acceptance testing stages. The goal is to determine whether or not the visible design and aesthetics of an application meet the intended workflow for various processes, such as logging into an application. Usability testing is a great way for teams to review separate functions, or the system as a whole, is intuitive to use.

7.9 Compatibility Testing

Compatibility testing is used to gauge how an application or piece of software will work in different environments. It is used to check that your product is compatible with multiple operating systems, platforms, browsers, or resolution configurations. The goal is to ensure that your software's functionality is consistently supported across any environment you expect your end users to be using.

7.10 Testing with Test complete

Test Complete is our robust automated GUI testing tool that excels in compatibility and integration testing. It helps QA teams create and run tests across desktop, mobile, and web applications – enabling testing professionals to speed up delivery cycles and improve software quality. Test complete comes with built-in support for various test environments, integrations to performance testing tools, as well as support for developer friendly SCMs, allowing you to seamlessly integrate it into your development process. Using Test Complete will enable you to build a robust testing framework that utilizes the broad spectrum of available software testing methodologies. In our proposed methodology, we have 90 percent training data and 10 percent testing data. Then we choose images randomly for testing data and test them to classify whether it is haze.

CHAPTER 8

CONCLUSION

8.1 Conclusion

This work proposes Arnold scrambling method and ECC which increases the security of scrambling technique. Only the genuine receivers with access to the shared key can extract both the message and the original cover image. The image is recovered with minimal losses. Thus the cryptographic security is enhanced.

8.2 Contribution

By using our proposed methodology we have enhanced the system security by using two level of security(Arnold transformation and Elliptical curve cryptography).If the first level of authentication somehow fails in some condition the malicious user will have to through second level i.e. ECC, therefore making the system very difficult for malicious users to hack.

8.3 Future Scope

The traditional scrambling algorithm based on Arnold transformation only applies to the square area, which is a big limitation. A multi-region algorithm for image scrambling encryption model is proposed, which splits the non-square image to multiple square regions, and scrambles each region.

REFERENCES

[1] Satish A , Erapu Vara Prasad, Tejasvi R , Swapna P , Vijayarajan R Image Scrambling through Two Level Arnold Transform Alliance International Conference on Artificial Intelligence and Machine Learning (AICAAM), April 2019.

[2] Dhananjay Santosh Waghulde , Dr. P. M. Mahajan A Review on Digital Image Scrambling Encryption Techniques Volume 5 Issue V, May 2017.

[3] Ohood S. Althobaiti1 and Hatim A. Aboalsamh An Enhanced Elliptic Curve Cryptography for Biometric January 2012.

[4] Siva Shankar S and A. Rengarajan DATA HIDING IN ENCRYPTED IMAGES USING ARNOLD TRANSFORM ICTACT JOURNAL ON IMAGE AND VIDEO PROCESSING, AUGUST 2016, VOLUME: 07, ISSUE: 01

[5] LipingZhang ShanyuTang HeLuo Elliptic Curve Cryptography-Based Authentication with Identity Protection for SmartGrids March 23,2016.

[6] A Fingerprint Biometric privacy using visual cryptography - M.Suganya and Dr.S. Suganya Volume-3, Issue-2, 2017

[7] S. EsaiPuvanesh S. Prakasam S. Vignesh G. Madasamy Raja SECURED TRANSMISSION OF BIOMETRIC CONTENT USING VISUAL CRYPTOGRAPHY Volume 119 No. 15 2018.