



School of Information Technology & Engineering

M. Tech Software Engineering

Final Report

Information and System Security (SWE3002)

**Title: Spoofing attack avoidance using Eye
movement based authentication system**

Submitted by

21MIS0401 – Nitheesh. T

21MIS0289 – Sabaregirivasan. M

21MIS0188 – Vijay Charan. K

21MIS0108 – Laxman Reddy. N

Faculty: Prof. Mangayarkarasi. R

Slot : A1+ TA1

SPOOFING ATTACK AVOIDANCE USING EYE MOVEMENT BASED AUTHENTICATION SYSTEM

ABSTRACT

As technology grows rapidly, the smartphone has become indispensable for transmitting private user data, storing the sensitive corporate files, and conducting secure payment transactions. However, with security research lagging, smartphones are extremely vulnerable to unauthenticated access. This paper presents an approach for an authentication method of people by using their eye movements. People's eye movements were recorded by using built in camera and compares it with the dataset and authenticate user to access the system.

I. INTRODUCTION

One of the security requirements for general terminal authentication systems is to be easy, fast and secure as people face, authentication mechanisms every day and must authenticate themselves using conventional knowledge based approaches like passwords. But these techniques are not safe because they are viewed by malicious observers who use surveillance techniques such as shoulder surfing (observation user while typing the password through the keyboard) to capture user authentication data. Also there are security problems due to poor interactions between systems and users. As a result, we proposed an eye tracking system, where the user can authenticate themselves as a correct person by expressing their eye movement to the authentication camera.

II. LITRATURE SURVEY

1. Person verification via eye movement-driven text reading model

The paper presents a reading-based eye movement biometrics model. The model is able to process passages of text and extract metrics that represent the physiological and behavioral aspects of the eye movements in reading. When tested on a database of eye movements from 103 individuals, the model yielded the Equal Error Rate of 10.2%. The proposed method performed better in the template-aging scenario than comparable eye movement-driven biometrics methods.

2. Eye movement analysis for human authentication: a critical survey, Pattern Recognition Letters

This paper addresses the active and dynamic nature of biometrics, in general, and gaze analysis, in particular, including motivation and background. The paper includes a critical survey of existing gaze analysis methods, challenges due to uncontrolled settings and lack of standards, and outlines promising future R&D directions. Criteria for performance evaluation are proposed, and state-of-the art gaze analysis methods are compared on the same database set. Performance improvement would come from richer stimuli including task dependent user profiles, with applications going much beyond identity management to include personalized medical care and rehabilitation, privacy, marketing, and education.

3. Evaluation of eye-gaze interaction methods for security enhanced PIN-entry

In this work, they evaluate three different eye gaze interaction methods for PIN-entry, all resistant against these common attacks and thus providing enhanced security. Besides the classical eye input methods they also investigate a new approach of gaze gestures and compare it to the well-known classical gaze-interactions. The evaluation considers both security and usability aspects. Finally they discuss possible enhancements for gaze gestures towards pattern based identification instead of number sequences.

4. ColorPIN: Securing PIN entry through indirect input

In this paper, the authors present ColorPIN, an authentication mechanism that uses indirect input to provide security enhanced PIN entry. At the same time, ColorPIN remains a one-to-one relationship between the length of the PIN and the required number of clicks. A user study showed that

ColorPIN is significantly more secure than standard PIN entry while enabling good authentication speed in comparison with related systems.

5. Eyepass - eye-stroke authentication for public terminals

In this paper, they presented EyePass, an authentication mechanism based on PassShape and eye-gestures that has been created to overcome these problems by eliminating the physical connection to the terminals. EyePass additionally assists the users by providing easy-to-remember PassShapes instead of PINs or passwords. They present the concept, the prototype and the first evaluations performed. Additionally, the future work on the evaluation is outlined and expected results are discussed.

Related Works:

There has been significant research in the field of spoofing attack avoidance using eye movement-based authentication systems. Some related works are:

"EyeSpy: Exploring the Feasibility of Eye-Based Verification for Mobile Authentication" by Guo et al. (2018). This study explores the feasibility of using eye-based verification as a secure and convenient authentication method for mobile devices.

"EyeVerify: Eye Movement Biometrics for Authentication and Authorization" by Bulling et al. (2012). This research proposes a novel eye movement-based biometric authentication system that is resistant to attacks such as replay and impersonation attacks.

"A Study on Spoofing Attacks in Eye Movement-Based Authentication" by Kim et al. (2018). This study investigates the vulnerability of eye movement-based authentication systems to different types of spoofing attacks and proposes a robust authentication method that is resistant to these attacks.

"Secure Mobile Authentication Using Eye Movements" by Xu et al. (2014). This research proposes a secure mobile authentication system that uses eye movements to authenticate users and is resistant to different types of attacks such as shoulder surfing and video-based attacks.

"Eye Movement Biometrics: A Survey" by Bulling et al. (2014). This survey provides an overview of eye movement biometrics and the different types of eye-based authentication systems that have been proposed in the literature.

III. EXECUTION CODE

```
from PySimpleGUI.PySimpleGUI import
Window import cv2
import numpy as np
import os
import time

from numpy.lib.type_check import
_getmaxmin from gaze_tracking import
GazeTracking import PySimpleGUI as sg
from collections import OrderedDict

starttime=0.0

def starttimes(text):
    global starttime
    starttime=0.0
    starttime=time.time()
    print(text)

def login(flag):
    sg.theme("LightGreen")

    # Define the window
    layout layout = [
        [sg.Text("Window Main", size=(80, 1), justification="center")],
        [sg.Image(filename="", key="-IMAGE-")],
        [sg.Radio("None", "Radio", True, size=(10, 1))],

        [
```

```

        sg.Button("Blur", "Radio", size=(10, 1), key="-BLUR-"),
        sg.Slider(
            (1, 11),
            1,
            1,
            orientation="h",
            size=(40, 15),
            key="-BLUR SLIDER-",
        ),
    ],
    [
        sg.Button("Hue", "Radio", size=(10, 1), key="-HUE-"),
        sg.Slider(
            (0, 225),
            0,
            1,
            orientation="h",
            size=(40, 15),
            key="-HUE SLIDER-",
        ),
    ],
    [
        sg.Button("Enhance", "Radio", size=(10, 1), key="-ENHANCE-"),
        sg.Slider(
            (1, 255),
            128,
            1,
            orientation="h",
            size=(40, 15),
            key="-ENHANCE SLIDER-",
        ),
    ],
    [sg.Button("Exit", size=(10, 1))],
]

# Create the window and show it without the plot
window = sg.Window("Auth++", layout, location=(400, 100))
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('c:/Users/Abdur Rahman Adeel/Desktop/FaceAuth/trainer/trai
ner.yml')
cascadePath = "c:/Users/Abdur Rahman
Adeel/Desktop/FaceAuth/haarcascade_fr ontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath)
font = cv2.FONT_HERSHEY_SIMPLEX
gaze = GazeTracking()

#iniciate id counter

```

```

id = 0
usernames=OrderedDict()
f = open('trainer/file.txt', 'r')
read=f.read()
usernames =
eval(read) f.close()

if flag==1:
    frr=OrderedDict()
    f1 = open('trainer/fileone.txt', 'r')
    read=f1.read()
    frr = eval(read)

    f1.close()

# Initialize and start realtime video capture
cam = cv2.VideoCapture(0,cv2.CAP_DSHOW)
# cam.set(3, 640) # set video width
# cam.set(4, 480) # set video height

# Define min window size to be recognized as a
face minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)

text=""

texttwo=""

textthree=""

vtop=vbottom=vleft=vright=0

registerb=False
loginb=False

xy=0
xz=0
length=0
blinkcount=0
arr=[0] * 4
counterl=0
arrl=[0] * 4
counter=0
verifynum=0

endtime=0

```

```

totaltime=0

tems=0.0
tt=0
id1=0

if flag==2:
    registerb=True
    loginb=False
    blinkcount=0
    print("Started Monitoring for Registration")
if flag==1:
    registerb=False
    loginb=True
    blinkcount=0
    print("Started Monitoring for login")
while True:
    if counter>=4:
        break
    if counter1>=4:
        break

    event, values = window.read(timeout=20)
    if event == "Exit" or event == sg.WIN_CLOSED:
        break

    tems=time.time()-float(starttime)
    if tems<1.5 and tems>1.2:
        starttimes(text)
    ret, img =cam.read()
    gaze.refresh(img)
    img= gaze.annotated_frame()

    if values["-BLUR-"]:
        img = cv2.GaussianBlur(img, (21, 21), values["-BLUR SLIDER-"])
    elif values["-HUE-"]:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
        img[:, :, 0] += int(values["-HUE SLIDER-"])
        img = cv2.cvtColor(img, cv2.COLOR_HSV2BGR)
    elif values["-ENHANCE-"]:
        enh_val = values["-ENHANCE SLIDER-"] / 40
        clahe = cv2.createCLAHE(clipLimit=enh_val, tileGridSize=(8, 8))
        lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
        lab[:, :, 0] = clahe.apply(lab[:, :, 0])
        img = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(

```



```

        gray,
        scaleFactor = 1.2,
        minNeighbors = 5,
        minSize = (int(minW), int(minH)),
    )

    if gaze.is_left():
        text = "Looking left"
        if tems>0.5 and tems<1.0:
            if counter<4:
                if registerb:
                    arr[counter]="left"
                    counter=counter+1
                    starttimes(text)
            if counterl<4:
                if loginb==True:
                    arrl[counterl]="left"
                    counterl=counterl+1
                    starttimes(text)

    elif gaze.is_top():
        text = "top"
        if tems>0.5 and tems<1.0:
            if counter<4:
                if registerb:
                    arr[counter]="top"
                    counter=counter+1
                    starttimes(text)
            if counterl<4:
                if loginb==True:
                    arrl[counterl]="top"
                    counterl=counterl+1
                    starttimes(text)

    elif gaze.is_right():
        text = "Looking right"
        vright=vright+1
        if tems>0.5 and tems<1.0:
            if counter<4:
                if registerb:
                    arr[counter]="right"
                    counter=counter+1
                    starttimes(text)
            if counterl<4:
                if loginb==True:
                    arrl[counterl]="right"
                    counterl=counterl+1
                    starttimes(text)

```

```

elif gaze.is_center():
    text = "Looking center"

if gaze.is_blinking():
    texttwo = "Blinking"
    blinkcount=blinkcount+1
    starttimes(texttwo)
elif gaze.is_blinking()==False:
    texttwo="Not blinking"

for(x,y,w,h) in faces:

    cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)

    id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

    if (confidence < 100):

        id1=id
        id = usernames[str(id)]
        confidence = " {0}%".format(round(100 - confidence))

    else:
        id = "unknown"
        confidence = " {0}%".format(round(100 - confidence))
        text="Unidentified face"

    cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
    cv2.putText(img, str(confidence), (x+5,y+h-
5), font, 1, (255,255,0), 1)
    cv2.putText(img, str(text), (10, 65), cv2.FONT_HERSHEY_DUPLEX,
1.2, (0,0,0), 2)
    cv2.putText(img, str(texttwo), (10, 30), cv2.FONT_HERSHEY_DUPLEX,
1.2, (0,0,0), 2)

    imgbytes = cv2.imencode(".png", img)[1].tobytes()
    window["-IMAGE-"].update(data=imgbytes)

    k = cv2.waitKey(10) &
0xff if k == 27:
        break

if flag==1:

```

```

        if counter1==4:

            for countss,xx in enumerate(frr[id1]):
                print("frr : "+str(frr[id1][countss])+" arr1 : "+str(arr1[countss]))

                if(frr[id1][countss]==arr1[countss]):
                    verifynum+=1

            if verifynum==counter1:
                sg.popup("Login Successful")
                print("welcome!!")
            else:
                print("Wrong secure code!!")

print("\n Exiting Program and cleanup stuff")

cam.release()

if flag==2:
    gesturecode=OrderedDict()
    try:
        f = open('trainer/fileone.txt', 'r+')
        read=f.read()
        gesturecode = eval(read)
        gesturecode[id1]=arr
        # print(usernames[1])
        f = open('trainer/fileone.txt', 'w+')
        f.write(str(gesturecode)) f.close()

    except Exception as e:
        print(e)
        f = open('trainer/fileone.txt', 'w+')
        gesturecode[id1]=arr
        f.write(str(gesturecode)) f.close()

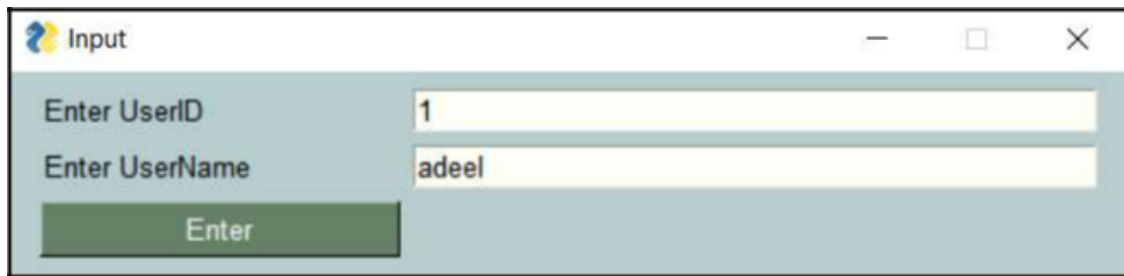
    print("Successfully Registered Gesture")

window.close()

```

IV.EXPERIMENTAL RESULTS

1. Registration Process:



A screenshot of a Windows application window titled "Input". The window has a light blue background. It contains two text input fields. The first field is labeled "Enter UserID" and contains the number "1". The second field is labeled "Enter UserName" and contains the name "adeel". Below these fields is a green button labeled "Enter".

The Console interface once the training process completed.

```
[INFO] Exiting Program and cleanup stuff  
[INFO] Training faces. It will take a few seconds. Wait ...  
[INFO] 1 faces trained. Exiting Program
```

4. Login Process:



A screenshot of a Windows application window titled "Auth++". The window has a light blue background. It contains three green buttons arranged horizontally. The buttons are labeled "Register", "Register Eye Gesture", and "Login".

The console output once the user has given the wrong eye movement.

```
frr : right arrl : right
frr : right arrl : top
frr : left arrl : right
frr : left arrl : right
Wrong secure code!!

Exiting Program and cleanup stuff
█
```

V. SECURITY ANALYSIS:

Security Attack can occur at any time. Then implementations of a system which avoid those attacks are in need. Here we explained few major security attacks and how our system avoid those attacks.

a. Spoofing/Masquerade Attack

A Masquerade attack is an attack that uses a fake identity, such as a network identity, to gain unauthorized access to personal computer information through legitimate access identification.

Justification:

As our system store the information of user while registering, authentication for unknown users is not possible.

b. Replay Attack

Replay attack involves the passive capture of a data unit and its subsequent retransmission to produce an authorized effect.

Justification:

As we already store our face identity in the registration phase, though the attacker stole the User Id and Password, he need the user to pass the first authentication process(Face authentication). So Replay attack is not possible.

c. Phishing Attack

Phishing is a type of social engineering attack often used to steal user data, including login credentials and credit card numbers. It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message.

Justification:

Two-factor authentication(Face authentication & Eye gesture authentication) is the most effective method for countering phishing attacks, as it adds an extra verification layer when logging in to sensitive applications.

d. Brute Force Attack

Allows an attacker to guess a person's user name, password, credit card number, or cryptographic key by using an automated trial and error process.

Justification:

As we discussed in attack, even though the attacker tries to authenticate using User Id and password, he can't pass through the eye movement verification process, where he need exact eye movement given by the original user while registering which is the password to authenticate user again. So Brute Force attack is also not possible.

VI. CONCLUSION AND FUTURE WORKS

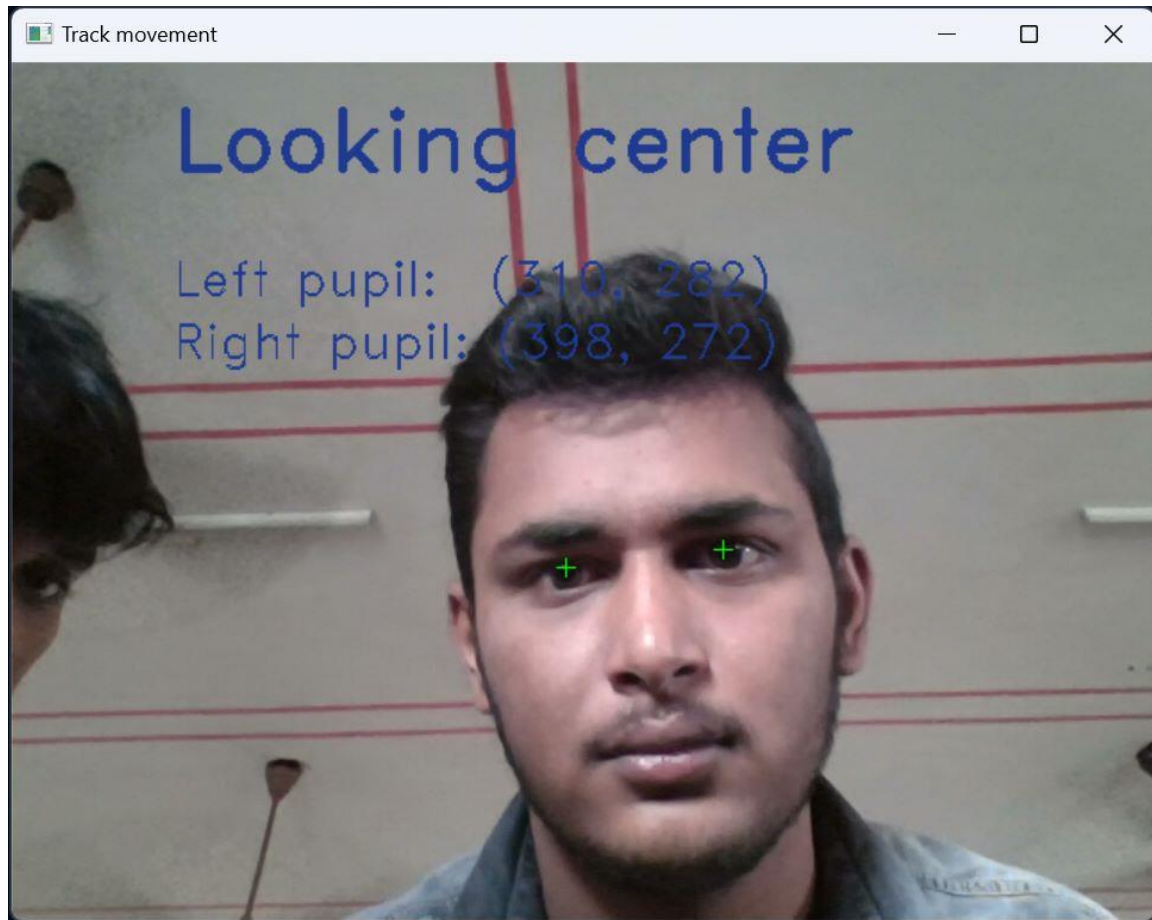
In this paper, we proposed an eye tracking system, where the user can authenticate themselves as a correct person by expressing their eye movement to the authentication camera for secured interaction with system environment. In future work, we plan to evaluate shape-based passwords on their appropriateness for this specific task.

VII. REFERENCES

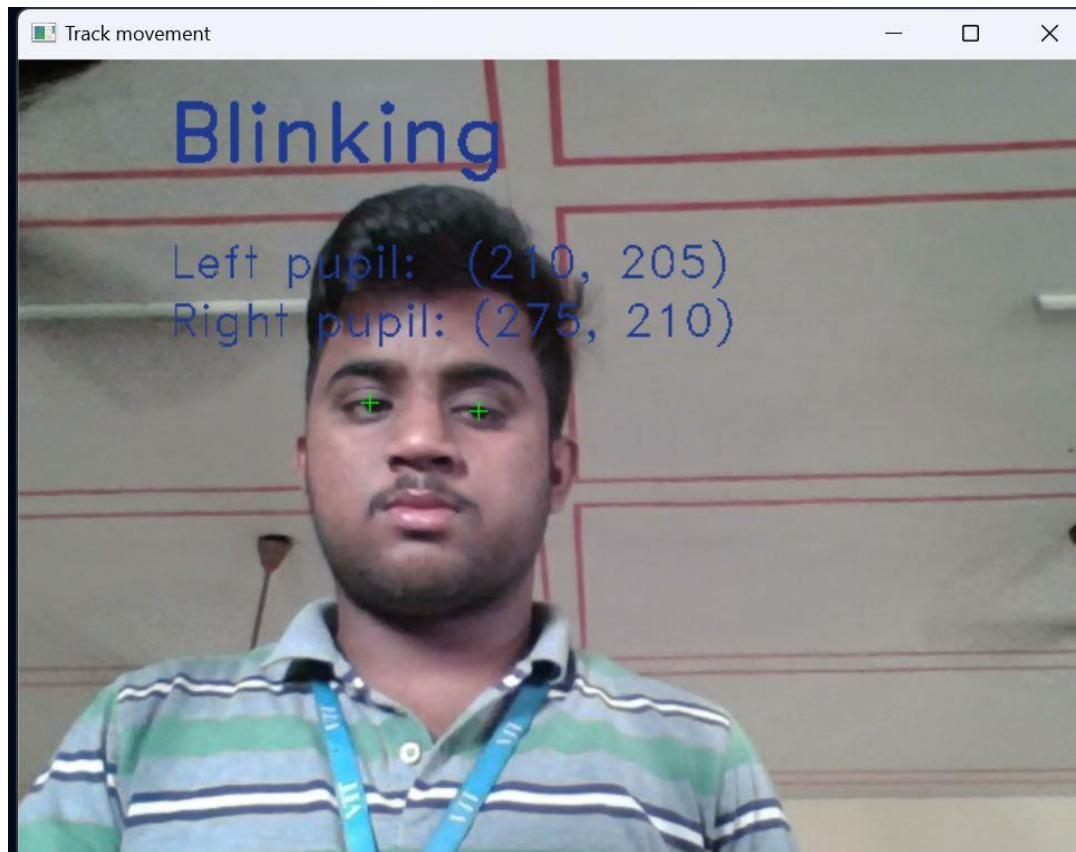
- [1] E. R. Abdulin and O. V. Komogortsev, "Person verification via eye movement-driven text reading model," 2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS), 2015, pp. 1-8, doi: 10.1109/BTAS.2015.7358786.
- [2] Chiara Galdi, Michele Nappi, Daniel Riccio, Harry Wechsler,"Eye movement analysis for human authentication: a critical survey,Pattern Recognition Letters, Volume 84,2016,Pages 272-283, ISSN 0167-8655, <https://doi.org/10.1016/j.patrec.2016.11.002>.
- [3] Alexander De Luca, Roman Weiss, Heiko Drewes "Evaluation of eye-gaze interaction methods for security enhanced PIN-entry" 2007, DOI: 10.1145/1324892.1324932

- [4] Alexander De Luca, Katja Hertzschuch, Heinrich Hussmann, "ColorPIN: Securing PIN entry through indirect input", 2010, DOI: 10.1145/1753326.1753490
- [5] Alexander De Luca, Roman Weiss, Heinrich Hussmann, Xueli An, "Eyepass - eye-stroke authentication for public terminals, January 2008, DOI: 10.1145/1358628.1358798

Registration Process:



Login verification interface



The interface output once the four-eye movement has been verified successfully



Login Process:

