Facial Recognition Presentation Attack Defense Through Liveness Detection

Nathan Rice

March 2022

Abstract

Biometric use for authentication and identification of individuals has become a common occurrence in the modern world. Facial recognition is one of the most commonly used biometrics. Facial recognition systems are commonly used in border control to identify travelers, in surveillance systems to identify criminals, in banks to identify users of ATMs, and in personal computers and smartphones for user authentication. Facial recognition systems have seen many technological advancements in recent years. However, there are vulnerabilities to these systems that are a point of concern. One such vulnerability is the presentation attack. The details of the presentation attack and the defense schemes to mitigate this vulnerability are the focus of this project. The objective is to propose viable solutions to implement presentation attack detection and defense. An end to end system implementing a liveness detection approach for presentation attack defense will be the result of this project. Lessons expected by tying this project with the course include knowledge of the facial recognition presentation attack, strategies for detection of the presentation attack, presentation attack defense algorithms and their feasibility in real-world applications, and the implications of facial recognition vulnerabilities on device security, and physical security in relation to user authentication, access control, and identity management.

Link to github project https://github.com/NateRice/FacialRecognitionPresentationAttackDefense

1 Introduction

The purpose of this project is to develop a system that detects a facial recognition presentation attack by determining the liveness of an input to a facial recognition system. The purpose of liveness detection in this case is to verify that the person is present at the input sensor at the time of the facial recognition capture. We wish to detect the liveness of the input to the system by detecting and counting the number of eye blinks. This is a type of passive liveness detection in which a challenge-response sequence is not needed and the liveness is determined by the observation of what is presented at the input sensor over

a short time period. This will allow for the detection of a presentation attack in which an attacker presents an image of a face to the input sensor. When considering the tradeoffs between convenience and security, this method has a noticeable advantage over the challenge-response method. The passive liveness detection method ensures a minimal inconvenience to the user. The user does not have to do anything but present themselves to the input sensor for 12 seconds. The method of eye blink counting using the eye aspect ratio requires a minimal amount of compute resources, especially when compared with complex and computationally heavy deep learning methods often used in other facial recognition liveness detection systems.

2 Problem Definition

In the scope of the C.I.A principles of cybersecutity, Confidentiality, Integrity, and Availability, this project falls under the integrity category. More specifically, end point integrity. The integrity of the input to the facial recognition system is the main concern. The vulnerability of facial recognition that this project is concerned with is a presentation attack, or spoofing attempt. A presentation attack is defined as "presentation to the biometric data capture subsystem with the goal of interfering with the operation of the biometric system"[ISO/IEC JTC 1/SC 37 Biometrics] by ISO/IEC 30107-1 term definition 3.5. This project investigates a facial recognition system presentation attack in which an attacker presents an artefact or presentation attack instrument to the input sensor with the goal of impersonating a person with authorized access to the system. For example, a print photo or an electronic display of a person's face. One way to determine if a person is present at the input sensor or if an image of a person is being presented to the sensor is through liveness detection. Liveness detection is defined by the "measurement and analysis of anatomical characteristics or involuntary or voluntary reactions, in order to determine if a biometric sample is being captured from a living subject present at the point of capture" [ISO/IEC JTC 1/SC 37 Biometrics definition 3.3]. This project will use an involuntary liveness detection method, eye blinking detection.

3 State of the Art and Challenges

Facial recognition systems are vulnerable to presentation attacks, also known as spoofing attacks. These systems are often unsupervised because of the automated nature of the task. For example, in border security, facial recognition kiosks provide an automated solution for traveler authorization while reducing staffing requirements. These unsupervised facial recognition systems are particularly vulnerable to presentation attacks.

A presentation attack occurs at the biometric data capture stage. An attacker's goal could be to conceal their identity or to be recognized as someone other than themselves, for example a person known to the system with autho-

rized access. In these cases an adversary will present an artifact of the authorized individual to the input sensor. An artifact is defined in ISO/IEC 30107-1 section 3.1 as an "artificial object or representation presenting a copy of biometric characteristics or synthetic biometric patterns"[ISO/IEC JTC 1/SC 37 Biometrics]. A facial recognition artifact can come in a few different forms. "Face artifacts can be easily generated simply by taking a photo of a legitimate user who is enrolled in the biometric system. Face artifacts are commonly generated using a photo print with a laser jet printer, a photo print with an inkjet printer, an electronic display of a photo or video of a face, or a 3D face mask and make-up"[Raghavendra Ramachandra and Christoph Busch]. It is not hard or expensive for an adversary to find a photo on a social media site of a known person and print that photo to be used in creating a face artifact for a presentation attack.

One countermeasure against a presentation attack is called liveness detection. Liveness detection is defined as "measurement and analysis of anatomical characteristics or involuntary or voluntary reactions, in order to determine if a biometric sample is being captured from a living subject present at the point of capture"[ISO/IEC JTC 1/SC 37 Biometrics]. One implementation of liveness detection for face recognition is the challenge-response method. This method of presentation attack detection can involve a call to action of the user and an expected response. For example a system could call for the user to turn his/her head to the left or right and the correct angle of head position will be expected by the system. This is called a voluntary response to indicate the liveness of the input on the system. Another type of liveness detection involves the detection of an involuntary response. In facial recognition an example of an involuntary response is blinking of the eyelids. It is possible to count the number of blinks as a person is being scanned by the system, and if the expected range of blinks is detected, the input may be validated. This can prevent a presentation attack where an adversary presents a static image to the input sensor. These types of challenge-response methods and liveness detection can verify a user's physical presence and mitigate some threats of presentation attacks. Other types of liveness detecion include image analysis for things like specular reflection properties of the image, image depth analysis for three dimensional detection, and image context analysis.

Another type of attack involves an adversarial image or tool that fools modern face detection algorithms. Deep learning methods of facial recognition are vulnerable to these types of attacks because they are specific to detecting examples that they have been trained on. When researchers and attackers form these adversarial devices, they are targeting the inner workings of the training algorithms with the goal of either going undetected or being falsely classified. One example of this method involves a hat with infrared lights attached to the brim. The researchers of this project claim they can trick a facial recognition algorithm into classifying a person as someone else by applying certain patterns of infrared light on the subject's face [Zhou et. al.]. Other adversarial methods involve certain patterns on a shirt, mask, or glasses. For example, researchers have identified some patterns of colors and shapes that when printed on the

frames of glasses and worn, can fool a facial recognition algorithm into classifying then as someone else[Vakhshiteh et. al.]. The exploration of adversarial tools to fool facial recognition algorithms is a state of the art area of research. Researchers are now looking into hardening these systems using deep learning by including these types of adversarial examples in the training set of the convolutional neural networks used in facial recognition models.

4 Work Conducted: classification or architecture (Model/Simulation/Prototype), tool used

If an adversary wishes to fool a facial recognition system into recognizing him/her as a known subject, the following steps may reasonably be attempted. The adversary acquires an image of a person known to have access to a system through facial recognition. This could be a printed social media profile photo or the adversary may have the photo on a handheld device. The adversary presents the facial artifact of the legitimate user to the facial recognition system input data sensor with the goal of gaining access to a system without giving up their identity. It is reasonable to expect that the adversary will not be seen presenting the image to the sensor because most of these systems are unsupervised due to the automated nature of the process, and the action could be quite discreet using a smart phone or even a small picture like an ID card.

The system uses passive liveness detection to verify the expected user's physical presence. If the expected response is not validated, the system denies access to the user and captures the evidence. If the expected response is confirmed, the user is granted access to the system. Refer to figure 1 for a flowchart of the process.

The type of passive liveness detection used in this project is the detection of eye blinking. The eyes are first localized in the image and tracked using a facial landmark detection model[Pretrained Facial Landmark Model D-Lib]. The specific points on the face detected by this pretrained model can be seen in figure 2. Once the eyes have been detected, the eye blinks can be counted by calculating the eye aspect ratio. The eye aspect ratio is the ratio of the horizontal distance of the eye to the vertical distance of the eye. The eye aspect ratio is mostly constant while the eye is open and converges to zero when the eye blinks. See figures 3 and 4 from Real-Time Eye Blink Detection using Facial Landmarks by Tereza Soukupova and Jan Cech. When the eye aspect goes below a certain threshold for a number of frames in the input video feed, a blink is counted. An artifact being used as input to a facial recognition system could falsely trigger a blink count using this method depending several factors such as lighting conditions, the angle of the photo, and the frame rate of the video feed. For this reason, an interval of acceptable blinks per second is used to classify an input as valid or invalid. After the input is presented to the sensor for 12 seconds, the blinks are counted and the blinks per second metric is used to determine the liveness. I have defined an interval of less than 0.16 blinks

per second or greater than or equal to 1.25 blinks per second as invalid. This flags a input with less than two blinks or more than 15 blinks in 12 seconds as a possible presentation attack. An individual blink is counted if the eye aspect ratio falls below 0.2 for 1 or more frames.

5 Results

The system was tested using real live input (refer to figure 5), a print photo (refer to figure 6), and a photo on a smart phone (refer to figure 7). The results were as expected, the system accepted the real live input and denied the photo input. The code to the program developed for the system is included in appendix A.

6 Conclusion

This research benefits all who are using facial recognition systems for user authentication or authorized access. Facial recognition systems are widely used, and the type of presentation attack discussed here is easy and inexpensive to perform. The solution proposed here does not present too much of an inconvenience to the user and is inexpensive to implement. Therefore, this presentation attack defense is a feasible solution. The system successfully detects a presentation attack using a still image. Future work could include a hybrid model that uses multiple methods of liveness detection to provide a more robust solution.

7 References

Eye blink detection with OpenCV, Python, and dlib. Adrian Rosebrock. pyimagesearch.com. https://pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/

Facial Point Annotations. Intelligent Behaviour Understanding Group (iBUG), Department of Computing, Imperial College London. https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/

ISO/IEC JTC 1/SC 37 Biometrics. ISO/IEC 30107-1:2016. Information Technology - Biometric Presentation Attack Detection - Part 1: Framework. International Organization for Standardization. info page

download link

Pretrained Facial Landmark Model D-Lib https://github.com/davisking/dlib-models/blob/master/shape$_p redictor_6 8_f ace_l andmarks.dat.bz 2$

Raghavendra Ramachandra and Christoph Busch. 2017. Presentation attack detection methods for face recognition systems: A comprehensive survey. ACM Comput. Surv. 50, 1, Article 8 (March 2017), 37 pages. DOI: http://dx.doi.org/10.1145/3038924 link to paper

Real-Time Eye Blink Detection using Facial Landmarks. Tereza Soukupova and Jan ˇCech .Czech Technical University in Prague https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf

Vakhshiteh et. al., Adversarial Attacks against Face Recognition: A Comprehensive Study. https://arxiv.org/ftp/arxiv/papers/2007/2007.11709.pdf

Zhou et. al., Invisible Mask: Practical Attacks on Face Recognition with Infrared. https://arxiv.org/pdf/1803.04683.pdf

A Facial Recognition Presentation Attack Defense: Involuntary Liveness Detection Through Eye Blinking: an application using Python, OpenCV, and D-Lib

```
# -*- coding: utf-8 -*-
"""
Created on Sat Apr  2 22:24:19 2022
Facial Recognition and Liveness Detection through Blink counts
Reference:
https://pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/

This program uses facial landmark detection to identify eyes of a face.
The eye aspect ratio is calculated.
The variable EYE_AR_THRESH = 0.2,
controls the threshold of the EAR that defines a blink.
The variable EYE_AR_CONSEC_FRAMES = 1,
is the number of frames the EAR must be below the threshold to count a blink
During the first 12 seconds of input from the laptop's camera,
if the frequency of blinks per second is below 0.16 or >= 1.25,
the input is denied and a possible presentation attack is detected.
That is, if there are less than 2 blinks counted in 12 seconds or
more than 14 blinks counted in 12 seconds,
the input considered a spoof attempt.

@author: Nathan Rice
"""
# import the necessary packages
from scipy.spatial import distance as dist
from imutils.video import FileVideoStream
from imutils.video import VideoStream
from imutils import face_utils
from timeit import default_timer as timer
import numpy as np
import argparse
```

```
import imutils
import time
import dlib
import cv2
import copy

def eye_aspect_ratio(eye):
        # compute the euclidean distances between the two sets of
        # vertical eye landmarks (x, y)-coordinates
        A = dist.euclidean(eye[1], eye[5])
        B = dist.euclidean(eye[2], eye[4])
        # compute the euclidean distance between the horizontal
        # eye landmark (x, y)-coordinates
        C = dist.euclidean(eye[0], eye[3])
        # compute the eye aspect ratio
        ear = (A + B) / (2.0 * C)
        # return the eye aspect ratio
        return ear

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
# #ap.add_argument("-p", "--shape-predictor", required=True,
# #     help="path to facial landmark predictor")
ap.add_argument("-v", "--video", type=str, default="",
    help="path to input video file")
args = vars(ap.parse_args())

# define two constants, one for the eye aspect ratio to indicate
# blink and then a second constant for the number of consecutive
# frames the eye must be below the threshold
EYE_AR_THRESH = 0.2
EYE_AR_CONSEC_FRAMES = 1
# initialize the frame counters and the total number of blinks
# number of successive frames that have an eye aspect ratio
# less than EYE_AR_THRESH
COUNTER = 0
# number of blinks that have taken place while the script has been running
TOTAL = 0

# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
print("[INFO] loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("./shape_predictor_68_face_landmarks.dat")

# grab the indexes of the facial landmarks for the left and
```

```
# right eye, respectively
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

# start the video stream thread
print("[INFO] starting video stream thread...")
#vs = FileVideoStream(args["video"]).start()
#fileStream = True
vs = VideoStream(src=0).start()
# start timer
start_time = timer()
# vs = VideoStream(usePiCamera=True).start()
#fileStream = False
time.sleep(1.0)


# loop over frames from the video stream
while True:
    # if this is a file video stream, then we need to check if
    # there any more frames left in the buffer to process
    #if fileStream and not vs.more():
    #    break
    # grab the frame from the threaded video file stream, resize
        # it, and convert it to grayscale
        # channels)
    frame = vs.read()
    frame = imutils.resize(frame, width=450)#width=450
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # detect faces in the grayscale frame
    rects = detector(gray, 0)
        # loop over the face detections
    for rect in rects:
        # determine the facial landmarks for the face region, then
        # convert the facial landmark (x, y)-coordinates to a NumPy
        # array
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        # extract the left and right eye coordinates, then use the
        # coordinates to compute the eye aspect ratio for both eyes
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        # average the eye aspect ratio together for both eyes
        ear = (leftEAR + rightEAR) / 2.0
        # compute the convex hull for the left and right eye, then
```

```
        # vizalize each of the eyes
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
        # check to see if the eye aspect ratio is below the blink
        # threshold, and if so, increment the blink frame counter
        if ear < EYE_AR_THRESH:
            COUNTER += 1
            # otherwise, the eye aspect ratio is not below the blink
            # threshold
        else:
            # if the eyes were closed for a sufficient number of frames
            # then increment the total number of blinks
            if COUNTER >= EYE_AR_CONSEC_FRAMES:
                TOTAL += 1

            #reset the eye frame counter
            COUNTER = 0

    # draw the total number of blinks on the frame along with
    # the computed eye aspect ratio for the frame
    cv2.putText(frame, "Blinks: {}".format(TOTAL), (10, 30),
    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 0), 2)
    cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 0), 2)


    # time duration reached, print accepted or denied
    end_time = timer() # stop timer
    duration = 0
    duration += end_time - start_time
    cv2.putText(frame, "Timer: {:.2f}".format(duration),(10,60),
    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 0), 2)
    blinks_per_sec = TOTAL/duration
    cv2.putText(frame, "BPS: {:.2f}".format(blinks_per_sec),(10,90),
    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 0), 2)
    if duration >= 12 and (blinks_per_sec >= 0.16  and blinks_per_sec < 1.25):
        cv2.putText(frame, "Input Accepted",(130,90),
        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
    if duration >= 12 and (blinks_per_sec < 0.16 or blinks_per_sec >= 1.25):
        cv2.putText(frame, "Input Denied", (130,100),
        cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)
        cv2.putText(frame, "Possible Presentaion Attack Detected!", (10,130),
        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

```
        # show the frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the 'q' key was pressed, break from the loop
        if key == ord("q"):
            print("Total Blinks: {}".format(TOTAL))
            # stop timer
            elapsed_time = end_time-start_time
            print("Elapsed time: {:.2f}".format(elapsed_time))
            print("Blinks per second: {:.2f}".format(blinks_per_sec))
            # results
            if blinks_per_sec < 0.16 or blinks_per_sec >= 1.25:
                print("Result: FAIL")
            if blinks_per_sec >= 0.16  and blinks_per_sec < 1.25:
                print("Result: PASS")
            break
# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

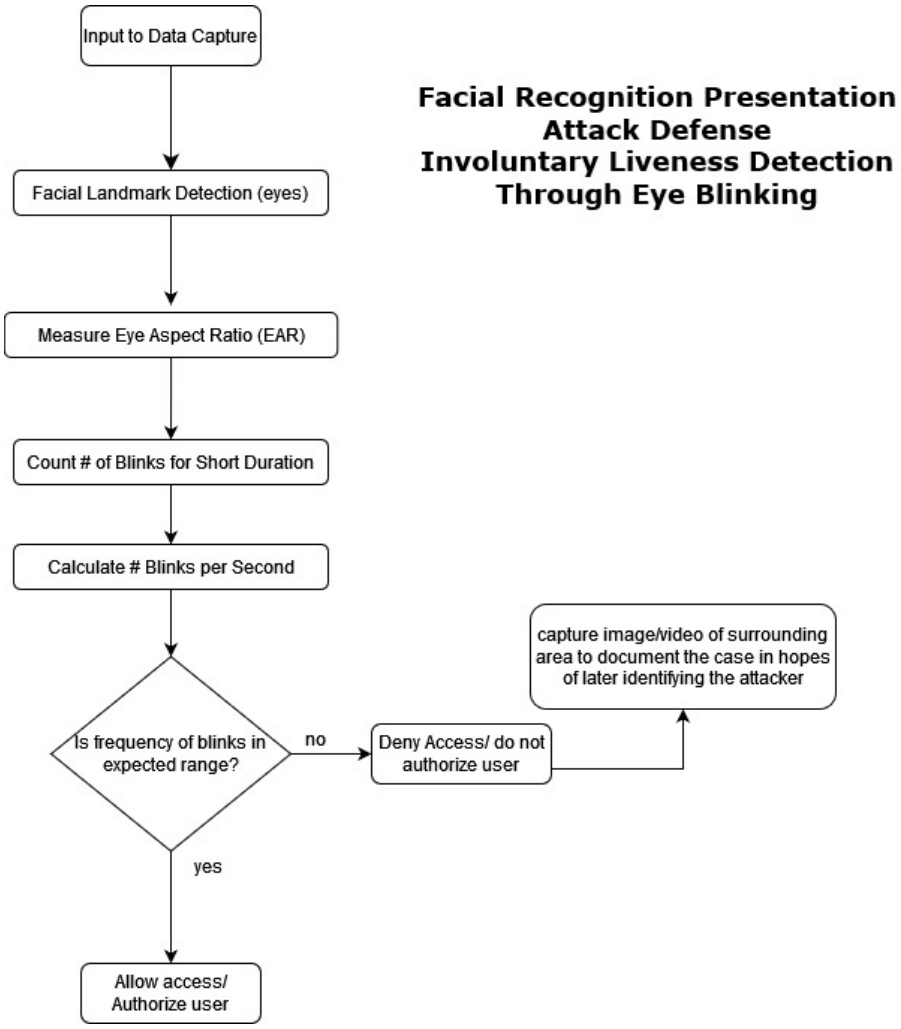

Figure 1: Presentation attack defense using passive method of liveness detection.

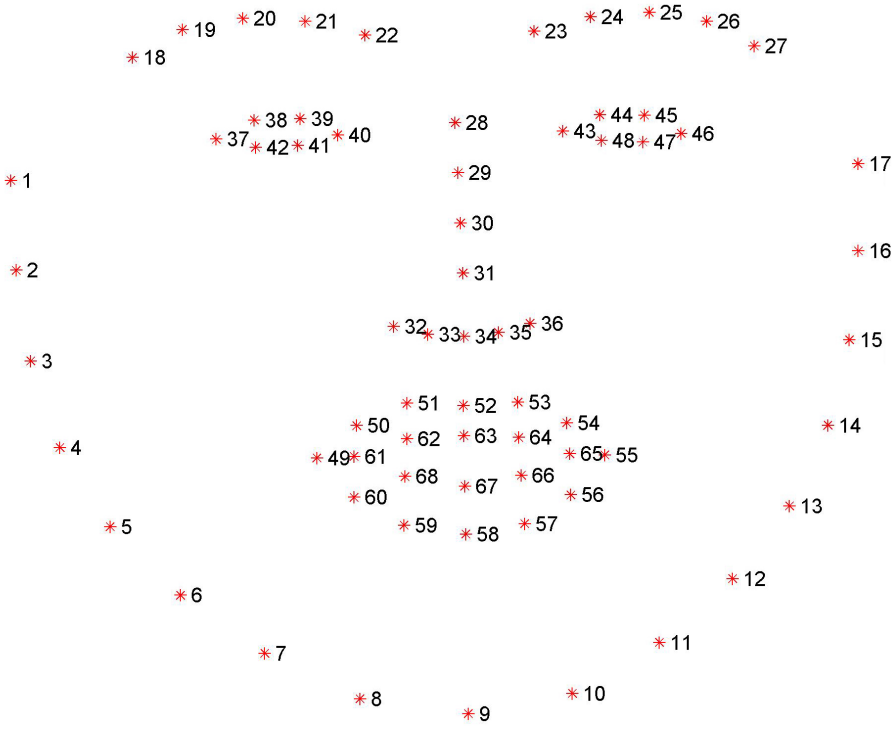

Figure 2: Facial Landmark Points. [Facial Point Annotations(iBUG)]

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Figure 3: Eye Aspect Ratio Formula Using Points in Fig 4

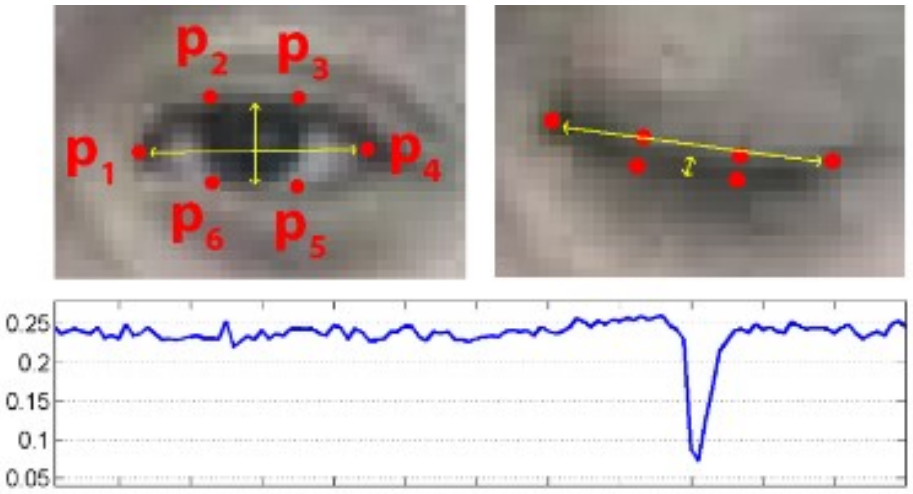

Figure 4: Eye Landmark Points - Detection of an Eye Blink

Figure 5: Screenshot of system test result of a real person at the input sensor.

Figure 6: Screenshot of system test result of a presentation attack using a print photo.

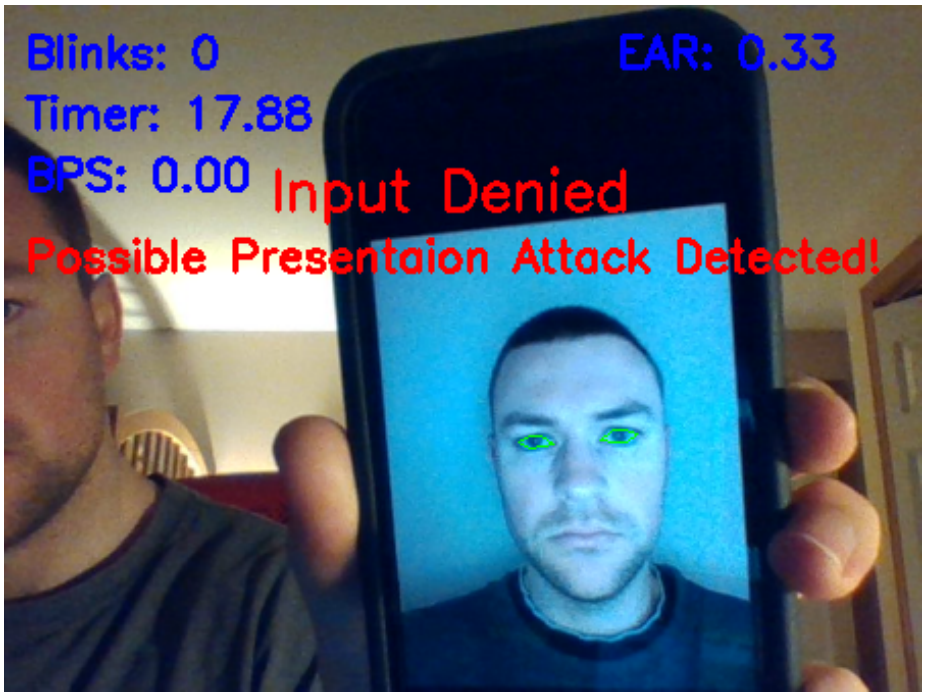

Figure 7: Screenshot of system test result of a presentation attack using a photo on a smartphone.