Blink as Security System

Ramya Danappa
University Of Texas Arlington
701 S Nedderman Dr, Arlington, TX 76019
ramyadanappa@gmail.com

Sanath Narasimhan
University Of Texas Arlington
701 S Nedderman Dr, Arlington, TX 76019
sanath.narasimhan@gmail.com

Abstract

Smartphones, laptops and many other electronics devices now-a-days have various techniques of user authentication and validation. Similarly blink detection is one of them. Human Blink can be use as one of the biometric authentications techniques as it is unique for each person. By using these blinking patterns, we can build an algorithm for blink authentication and validation. This requires having a device that can capture a user's eyes with individual cameras, a pair of glasses. This device can then be connected wirelessly with any device and can act like an authentication layer.

New user must set a user ID which will be any string of characters, followed by which we capture images of the eyes in the rest pose and the password. Here password is user's eye blinks. The blink is captured and stored in the file based on a set of calibrations for each user. These images and the calibrations will be used as input dataset for training the model. The trained model should able to validate the user when they try to login. Goal is to achieve less error rate and good security. Security is ensured while setting the password by calibrating the user's blink at different rates, slow, medium and fast paced. The password that will be set will a combination based on the three rates set during the calibration. This ensures that the blink cannot be completely replicated by anyone other than the user.

1. Introduction

This paper explores a new way of user authentication, in the past few decades security has become a prominent issue. The traditional techniques like entering a password through a real/virtual keyboard do lack the necessary measures to ensure high security and safety. For the past few decades this has been the problem tackled by many and there are quiet a few ideas that have taken a step towards a more secure approach, like the bio-metric user identification. Our aim is to eliminate room for snooping while a user is authenticating themselves from unwanted party.

1.1. Motivation

Conventional devices into which passwords are entered, and especially devices where passwords are entered into a display such as smartphones, ATMs, etc., have many characteristics which make passwords and other authentication data vulnerable when being entered.

For instance, the smartphone may reveal each character on the display for a second or two during entry. Additionally, when a key is pressed on the screen of a smartphone the user is often given a visible cue indicating that the particular character has been pressed. The visible cue can take many possible forms such as a magnification of the character on the keypad, changing the color of the character on the keypad, movement of the character selected, etc. These visible indications of which characters are being entered can enable someone to steal the password by simply reading the password as it is entered. Even when the actual entering of the password is not observed, the security of the password may still be degraded by residual marks left on the screen by the operator (e.g., "fingerprints"). Such fingerprints may reveal the password information entered by the location of the keys pressed or the path of a finger as it moved on the screen.

In addition, on a smartphone the displayed keyboard normally has significantly fewer keys than a conventional keyboard. Therefore, entry of the characters on the on-screen keypad divulges possible character positions. For example, with a simplified keyboard it is not difficult to determine which character has been pressed by observing the operator's hands and location on the display, which the operator presses. Further compounding this problem is that the keypad displayed may vary depending on the character type being entered. Such a feature can reveal when the user types a number, lowercase or uppercase character, symbol, etc, further increasing the ability of an onlooker to judge which character is being entered.

The environment in which a password is entered may further amplify the password's vulnerability to detection. Such environmental factors may be anything from a reflective surface to aid an on-looker, to security cameras, which

record the password as it is entered, etc.

All of these issues weaken the security provided by a password entered into a device such as a smartphone, ATM, etc.

Therefore, it is insecure (i.e., non-secure) to enter passwords onto a smartphone. As a result, a smartphone, or other display device, is rendered insecure which makes the smartphone undesirable to use in accessing sensitive data.

While the above problems have been described in terms of a smart phone, the basic problem also applies everything from tablets, laptops, computer keyboards, ATMs, etc., or any device where a code is manually entered.

2. Security system Overview

2.1. Data collection and Calibration

The approach to securing the password involves asking each new user to calibrate two different rates of blinking. The two types of blinks are captured separately.
To capture the blink we use the facial landmark detection to localize the eyes in a given frame from a local webcam, this is achieved with the help of HOG and Linear following [2] and [4]. We use a pre-trained model on every frame for retrieving the Eye Aspect Ratio (EAR) which is the ratio of the height of the eye to its width. From the landmarks detected we get 6 for each eye, using which we get 2 heights and one width (See Figure).

Figure 1: EAR open

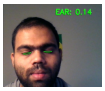

Figure 2: EAR closed

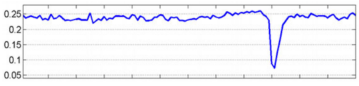

Figure 3: formulae

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

We compute for each eye the EAR and take their average, assuming the eyes are parallel to the camera. For collecting the user blink data we assume that any frame with EAR below 0.17 is a potential frame for blink. But we cannot rely on just this threshold as there may be quiet a few false positives. The built-in webcam has approximately a FSP of 30, based on this we define two variables to keep track of the number Inter blink frames and the number of frames a blink was detected for, that is average EAR for the frame is below 0.19.
To further avoid any mislabeled frame, we say that at least 2 frames must have passed before detecting a next blink and the number of frames the blink lasted for should be minimum of 3 and a maximum of 32 (any blink longer that 30 frames, 1 second is discarded).

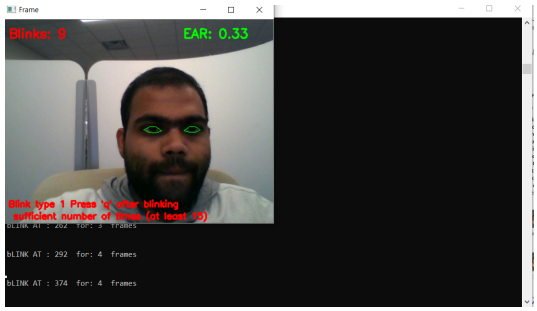

Figure 4: Blink Type 1 capture, In this case fast blinks with an average of 4 frames per blink

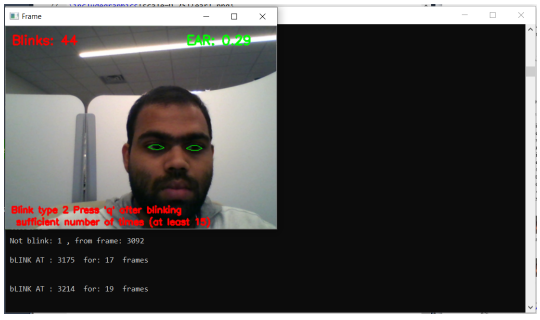

Figure 5: Blink Type 2 capture, In this case it is slow blink with an average of 17 frames per blink

2.2. Training the classifier

At least 15 blinks need to be detected for each type of the blink. The data is restructured such that for every frame we collect the average EAR's of 7 neighbouring frames on either side and the average of all 15 frames and features for building a good classifier.
This is then passed to a Linear SVM with C=1 making sure that it doesn't over-fit the data nor generalizing too much.

The classifier is evaluated with 40 percent of the data randomly sampled. In this case we need to make sure the precision is high as blinks are rare events compared to non blink frames. In order for the classifier to be robust make

```
Accuracy: 0.9552845528455285  Precision: 0.9183673469387755  Recall: 0.967741935483871
```

Figure 6: On an average the SVM classifier built has the following evaluation results

sure that there is enough light falling over the eyes and ensure that you are at least 20 centimeters from the camera.

2.3. Detecting and storing password

We record each blink by detecting the type simultaneously, each classifier analyses the frame and decides if it belongs to a type of blink. Once the password is finalized, an encoder dictionary is used to determine the mapping of sub-patterns of the blinks within the whole string.

```
Encode ={"1111":"A","1112":"B","1121":"C","1122":"D","1211":"E","1212":"F","1221":"G","1222":"H"
        ,"2111":"I","2112":"J","2121":"K","2122":"L","2211":"M","2212":"N","2221":"O","2222":"P","111":"Q"
        ,"112":"R","121":"S","122":"T","211":"U","212":"V","221":"W","222":"X","1":"Y","2":"Z"}
```

Figure 7: The keys correspond to sub patterns within the sequence of blinks, the values are their encoded form.

The password has to be of minimum length 8 and maximum length 64. We take the modulus of the length of entered blink sequence and handle each case. When remainder is 0, the whole sequence is divided into batches of 4. If the remainder is 1 or 2, for the last few blinks are mapped individually. If the remainder is 3 then we take sequences of 4 till only last 3 are left and encode the 3 separately.

```
{'fosho1': ['fosho1_t1.pickle', 'fosho1_t2.pickle', 'FO']}
```

Figure 8: The keys is the user name, first element of key is path for data of type1 blink, second element of key is path of data for type2 blink.

The actual blink sequence for user 'fosho1' is '12122221' but is stored as 'FO'

3. Result

We have achieved user authentication method by using sub sequence of blink patterns of user eye.When user enter the wrong sequences of blink patterns or if any other users try to imitate the user password. This model will not authenticate as valid user. Hence we can say it is more secured method of password.

In Figure 9, the When user try to enter the correct password by using the blink, then our method validate the patterns directly mapped with the blink sub sequence stored in database.

In figure 10, there user password is authenticated and login ed successfully.

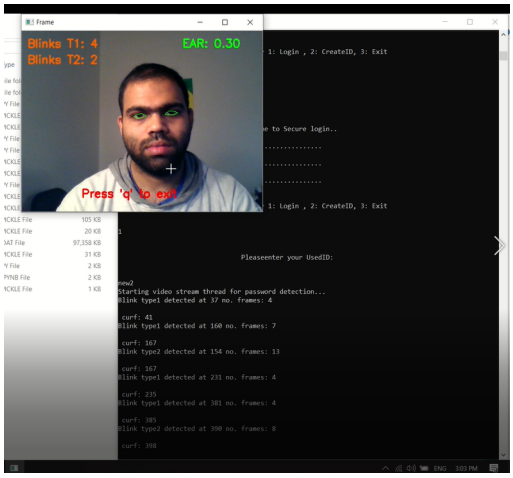

Figure 9: User Authenticating by providing blink Patterns

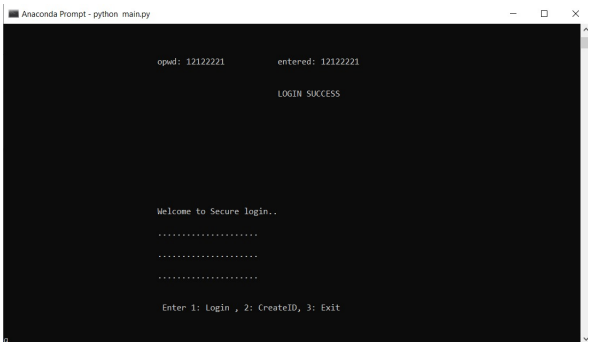

Figure 10: The correct eye blink validates and login successfully

4. Conclusion

To make things both interesting from an interaction perspective, to and make the password harder to crack by tracking eye-blinks, the authentication interaction can be modeled as a game (and even have levels), where, based on the skill level of the user, various parameters may be changed dynamically; for example, a pair of rapid blinks (instead of one) for a letter, changing the combination of L/R blinks for a letter, invoking the correct letter by providing (guessing) the blinks that will produce it, etc. If the system thinks that the user is taking too much time, or the user is in a hurry, the user can always reduce the level of difficulty of the game and enter the password. In addition, the speed of blinking, etc. may be customized for each user.

By this of method it is difficult to hacker to encrypt the password and hack the system.This method is highly secured because it is very hard to mimic the user blink. These blinks are directly mapped into characters based on blink patterns.

References

1. https://patents.google.com/patent/US9749137
2. https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/
3. https://www.frontiersin.org/articles/10.3389/fict.2018.00004/full
4. https://www.pyimagesearch.com/2015/02/02/just-open-sourced-personal-imutils-package-series-opencv-convenience-functions/

5. Contribution

Ramya Danappa: Blink detection through face landmark detection using dlib and imutlis.

Sanath Narashimhan: Collection of blink data, building SVM classifier using different blink rates

6. more references images

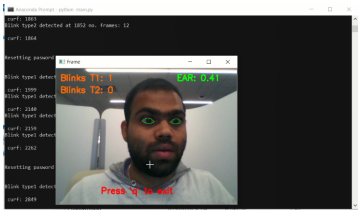

Figure 11: 1

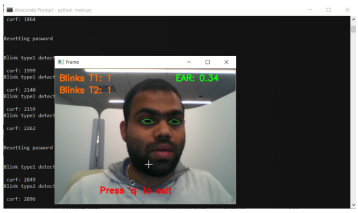

Figure 12: 2

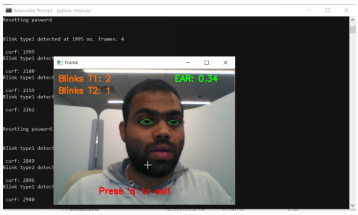

Figure 13: 3

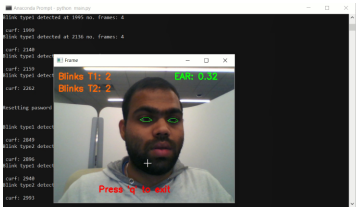

Figure 14: 4

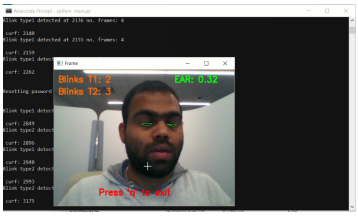

Figure 15: 5

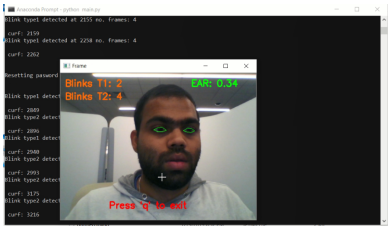

Figure 16: 6

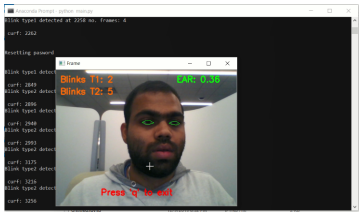

Figure 17: 7

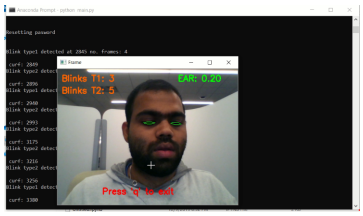

Figure 18: 8