DyGazePass: A Gaze Gesture-Based Dynamic Authentication System to Counter Shoulder Surfing and Video Analysis Attacks

Vijay Rajanna Adil Hamid Malla Rahul Ashok Bhagat Tracy Hammond
Sketch Recognition Lab, Dept. of Computer Science and Engineering, Texas A&M University
{vijay.drajanna, aaddil.hamid, rab248b, thammond}@gmail.com

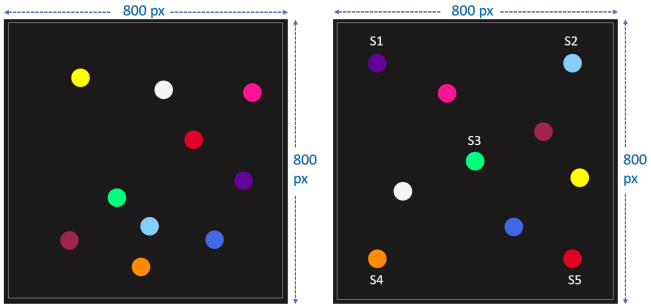
Abstract

Shoulder surfing enables an attacker to gain the authentication details of a victim through observations and is becoming a threat to visual privacy. We present DyGazePass: Dynamic Gaze Passwords, an authentication strategy that uses dynamic gaze gestures. We also present two authentication interfaces, a dynamic and a static-dynamic interface, that leverage this strategy to counter shoulder surfing attacks. The core idea is, a user authenticates by following uniquely colored circles that move along random paths on the screen. Through multiple evaluations, we discuss how the authentication accuracy varies with respect to transition speed of the circles, and the number of moving and static circles. Furthermore, we evaluate the resiliency of our authentication method against video analysis attacks by comparing it to a gaze- and PIN-based authentication system. Overall, we found that the static-dynamic interface with a transition speed of two seconds was the most effective authentication method with an accuracy of 97.5%.

1. Introduction

Shoulder surfing generally refers to unsolicited access to a user's confidential information (e.g., interests, hobbies, sexual preferences, login credentials, etc.) by an observer [6]. In this work, we focus on preventing shoulder surfing attacks on a knowledge-based authentication method, i.e., using passwords specifically in public and semi-private spaces. Keypad monitoring commonly occurs at public places like ATMs, kiosks, airport lounges, coffee shops, airplanes, and semi-private spaces like offices, aimed at stealing the login credentials of users [12]. A report on global visual hacking, presented by Ponemon Institute in 2016, found that in business office environments the attacks happen on laptops, tablets, smartphones, etc [7]. They conducted shoulder surfing attacks in eight countries, and a staggering 91% of visual attacks were successful, resulting in 613 units of breached data of various types [7]. Furthermore, 11% (69 units) of the breached data were login credentials. To prevent shoulder surfing, we focus on gaze-based authentication, which has been previously explored by [2, 3, 4, 9, 15]. The existing solutions are limited by low accuracy, the need for precise gaze input, accurate recall of the gestures by users, and susceptibility to video analysis attacks.

2018 IEEE 4th International Conference on Identity, Security, and Behavior Analysis (ISBA)
978-1-5386-2248-3/18/$31.00 ©2018 IEEE



(a) (b)

Figure 1: a) Dynamic authentication interface with 10 uniquely colored circles placed at random positions, b) Static-dynamic authentication interface comprising 5 static (S1, S2, S3, S4, S5) and 5 dynamic circles.

We present, a gaze gesture-based approach to addressing shoulder surfing on user authentication. The central idea of our authentication system is, the interface comprises of 10 uniquely colored circles which move simultaneously along random paths during an animation of N seconds. An animation is a time interval where all the circles move from their source to destination locations. Analogous to a four digit PIN, a user selects a set of four colors (out of 10) as her password. To authenticate, the user follows the path of the circle, colored with her password color, during an animation. This animation is repeated four times allowing the user to enter all four colors. For example, if the user's password is "red-blue-yellow-green" the user follows the red colored

circle during the first animation, blue during the second, and so on. For a successful authentication, the scan-path of the user's gaze should match with the path of the colored circle selected as the password, in each animation, for all the four animations. We hypothesize that people would be able to remember their password by associating the password colors with their favorite colors, the colors of the objects they frequently use (car, cloth), etc.

The two authentication interfaces we have developed are the "dynamic interface" shown in Figure 1a and the "static-dynamic interface" show in Figure 1b. Since we targeted our authentication system to be deployable at ATMs, kiosks, laptops, or in general, devices with smaller screens, we surveyed screen dimensions[1] [2] of ATMs by various vendors. There is no single standardized size of the ATM screen, but commonly used dimensions include 8", 10.1", 12.1", 15". We chose a median size of 11.5" which approximately translates to a dimension of 800×800 pixels on a screen with 98.44 PPI (screen size 1900×1200 px, 23") used in our experiments. We evaluated our solutions through a two phase user study. In the first phase, both dynamic and static-dynamic interfaces were tested for their accuracy under two animation speeds: 3 and 2 seconds. Since the static-dynamic interface was found to be a more practical solution, we tested its susceptibility to video analysis attack in the second phase.

2. Prior Work

Gaze as an input modality [13] has enabled various gaze-based authentication methods, against shoulder surfing, that can be classified across three broad categories.

2.1. Gaze- and PIN-based Authentication

Kumar et al. [9], presented "EyePassword," an authentication method where the user enters sensitive input like password or PIN by selecting from an onscreen keyboard using their gaze. Khamis et al. [8], presented "GazeTouchPass," that allows authentication on mobile phones though multiple switches between gaze and touch input modalities.

2.2. Gaze Gesture-based Authentication

De Luca et al. [4], presented "Eye-Pass-Shapes", where a user authenticates by drawing one of the eight gestures with their eye movements. The system evaluations showed that the eye gestures significantly increases security while being easy to use. Best et al. [1], presented a rotary interface for gaze-based PIN code entry. Their solution is based on the weighted voting scheme of numerals whose boundaries are crossed by the streaming gaze points. The authentication accuracy was found to be 71.16% with the PIN interface and 64.20% with the rotary interface.

[1] www.atmmarketplace.com [last accessed: 27th July 2017]

[2] www.ATMequipment.com [last accessed: 27th July 2017]

2.3. Gaze Pursuit-based Authentication

Rajanna et al. [14] presented an intelligent gaze gesture-based authentication system to counter shoulder surfing attacks. A user authenticates by her unique gaze patterns onto moving geometric shapes. The system achieved an authentication accuracy of 99% with true calibration and 96% with disturbed calibration. Vidal et al. [15] presented the idea and the design of authentication using eye pursuits. The authors proposed a pursuits-enabled screen that displays an animation of fishes swimming in the fish tank. The user can authenticate by looking at four specific fishes in the precise sequence. Cymek et al. [3], presented an authentication method, where the user follows the digits moving in vertical and horizontal directions to authenticate. The system accuracy was 97.57% in recognizing the digits entered.

As discussed before, the common limitation with gaze-based authentication systems is having a low accuracy. As evaluated by De Luca et al. [5], the error rate of various well known gaze-based authentication methods varied from 9.5% to 23.8%. Also, gaze authentication is susceptible to video analysis attacks as demonstrated in [2, 4]. Though Cymek et al. [3] and Rajanna et al. [14] had accuracies of above 95%, both were evaluated on larger screens. Furthermore, [14] was susceptible to video analysis attacks, and [3] was not evaluated for video analysis attacks. *The contribution of our work derives from trying to address these limitations.* Our authentication system achieves a *high accuracy* as we leverage template matching to recognize the gestures. Also, due to the dynamic nature of the interface, we show that our system is *not susceptible* to *single video* iterative attack, and has a *low success rate* with *dual video* iterative attack.

3. System Architecture

Our gaze gesture-based authentication system consists of two main modules: 1) Gaze Tracking Module, and 2) Authentication Engine. A working model of the system is depicted in Figure 2a.

Gaze Tracking Module: The system uses "The Eye Tribe" tracker[3], which is a table mounted eye tracking sensor that provides (X,Y) coordinates of the user's gaze on the screen at 60 Hz. For the eye tracker to work efficiently, the user is positioned such that the face is centered in front of the monitor at a distance of 45 - 75 cm.

Authentication Engine: The authentication engine is the central module that runs on a computer and receives (X,Y) gaze-coordinates from the eye tracker. This module is responsible for positioning the circles at random locations on the interface, and generating a random path for each circle. The module also implements the scan-path matching algorithm to authenticate a user.

[3] www.theeyetribe.com [last accessed: 27th July 2017]

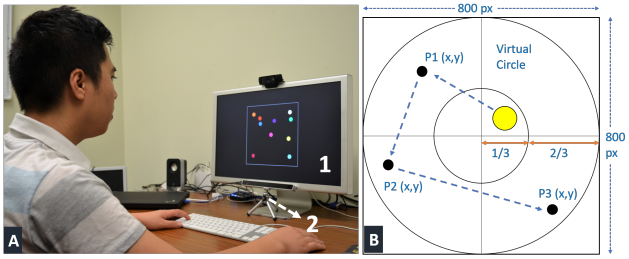


Figure 2: a) A user is authenticating by following the paths of four uniquely colored circles chosen as the password (1 - Authentication interface, 2 - Eye tracker), b) Distribution of Random Points: distribution of random points (P1, P2, P3) for the path of a circle (yellow). The random points are beyond 1/3 distance from the center along the radius of the virtual circular boundary.

3.1. Authentication Procedure

The authentication procedure comprises of two processes: 1) one-time password selection, and 2) password entry. The user selects four colors from a password selection window, one for each of the four animations, that form the password of the user. The user controls the authentication interface through a set of hot-keys on a keyboard: 'A' to initiate movement of the circles (start animation) and record gaze data, 'Z' to recover from user mistakes (blink, sneeze, losing the path, etc.) and discard recorded gaze data, and 'M' to submit the password after following four circles.

4. Authentication Interface Dynamics

Both the dynamic and static-dynamic interfaces have 10 colored circles and have dimensions of 800×800 px, but they differ in the number of moving circles during an animation. The two mechanisms that are common to both the interfaces are: 1) random point generation, 2) generation of the animation path and template for each colored circle.

4.1. Random Point Generation Algorithm

To generate n number of random points that are uniformly distributed inside a circle with radius R_c and area A, we employ the method proposed by Leon-Garcia et al. [10]. The joint probability distribution function (PDF) of the random points inside the circle, i.e., the joint distribution of random variable **X** and random variable **Y** representing the **x** and **y** coordinates of a random point is given by:

$$f_{X,Y}(x,y) = \begin{cases} \frac{1}{A} = \frac{1}{\pi R_c^2} & x^2 + y^2 \leq R_c^2 \\ 0 & \text{otherwise} \end{cases}$$

4.1.1 Initial Points for Circles

The initial locations of the circles on the interface are random but uniformly distributed within the radius of 100 pixels from the center of the interface (with dimensions of 800×800). To generate the points, we use the Random Point Generation Algorithm discussed in section 4.1 with radius $R_c = 100$ pixels for each of the colored circle. We are using a small radius to make the initial positions of the circles closer which makes video analysis attacks difficult.

4.2. Generating Animation Paths and Templates

Once the initial points are generated for the 10 circles, we generate a random path for each circle. Each random path is a set of three points that the circle traverses through from its initial point. To generate the three points for a random path, we use the same method of generating uniformly distributed random points discussed in section 4.1 with a radius of $R_c = 400$. Additionally, we constrain the three points to be beyond one-third of the distance from the center as shown in Figure 2b. Based on the interface dimension, duration of animation, and the sampling frequency of the eye tracker, we have established empirically that the template path should be made up of 300 points that are equally distributed along its path relative to the length of each line segment. Hence, all the points on a path obtained from the above computation are stored as the template path for matching against a scan-path.

4.3. Scan-path Matching Algorithm

We match the user's scan-path against a circle's traversed path through the "Template Matching" algorithm, where we compute the root-mean-square distance of the candidate path (user's scan-path) from all the template paths (circles' traversed paths). The template path of a circle that is at the least root-mean-square distance from the candidate path is chosen as the circle (color) followed by the user. Our template matching algorithm is based on $1 [16], but we perform only sampling, and calculate the average distance between the two paths.

Sampling: We down-sample both the candidate path and the template paths to N=64 points, because of two reasons. First, sampling reduces the noise in the scan-path due to inherently jittery eye movements, and approximates the path to a good extent. Second, down-sampling reduces the computation required during the matching phase. Figure 3 shows the sampling phase of a scan-path.

Matching: To compute the average distance between a candidate path and a template path with N points, as shown in Figure 3, we use Equation 1,

$$\Delta DT = \frac{\sum_{p=1}^{N} \sqrt{(C[p]_x - T[p]_x)^2 + (C[p]_y - T[p]_y)^2}}{N} \quad (1)$$

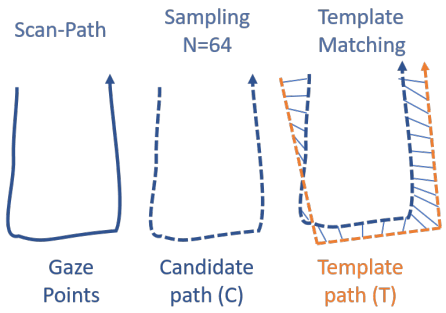


Figure 3: Sampling and Template Matching: a user's scan-path is sampled to N=64 points (candidate path), and matched against paths of all the circles (template paths).

where p is a point on path, C - candidate path, T - template path, and ΔDT - average distance to template.

5. Authentication Interfaces

5.1. Dynamic Authentication Interface

The dynamic authentication interface comprises of 10 uniquely colored circles that are distributed randomly on the interface. Most importantly, the circles traverse along random paths during an animation, and once they reach their final locations at the end of the animation, they interchange their positions in a random fashion.

5.2. Static-Dynamic Authentication Interface

We further modified the dynamic interface to develop the static-dynamic interface for two reasons. First, although the dynamic interface with 10 moving circles introduces enough randomness to prevent both shoulder surfing and video analysis attacks, a few users were overwhelmed by the visual cluttering of the interface as we found during the pilot studies. This means, although the user can start following a circle, once all the circles come closer or overlap during an animation, the user may lose sight of the circle, leading to recognition error. Second, since all the 10 circles move within a space of 800×800 px, two random paths might be similar which again leads to recognition failures that negatively affect the accuracy.

Considering these factors, we designed the static-dynamic authentication interface in such a way that only 5 out of the 10 circles move during an animation and the other 5 circles remain static. However, a dynamic (moving) circle on the current animation can continue to be a dynamic circle or become a static circle on the subsequent animation; the same is true for a static circle. Hence while authenticating, the user mostly ends up following a few dynamic circles and focusing on a few static circles. Since random placement of the static circles may bring two circles closer and result in recognition errors, we fixed the locations for static circles (Figure 1b) along the virtual circular boundary at angles 45°, 135°, 225°, 315°, and at the center of the rectangle.

5.2.1 Scan-path and Fixation Matching

Unlike the dynamic interface, we need to first distinguish if the user followed a circle or fixated on a circle in the static-dynamic interface. To accomplish this, at the end of every animation, we compute the length of the user's scan-path. If the length of the scan-path is above the dispersion threshold ($PathLength > d_{th} = 300$ pixels), we use the scan-path matching algorithm. However, if the length of the scan-path is below the dispersion threshold ($PathLength < d_{th} = 300$ pixels), we use the centroid method (Figure 4a) to recognize the target circle on which the user was fixated.

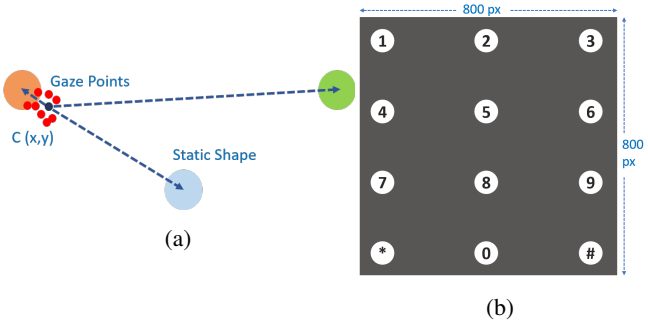


Figure 4: a) Centroid Method: to recognize the static circle focused on by the user, the centroid (c(x,y)) of the gaze points is found and distances to all the static shapes are calculated to find the least distance, b) PIN Interface with 10 digits spaced evenly.

6. Evaluation and Results

We recruited 20 participants (16 male, 4 female) whose ages ranged from 18 to 31 ($\mu_{age} = 23.15$). The participants entered passwords on both dynamic and static-dynamic interfaces, and the order of the interface presented to the participants was counterbalanced. Before the study, each participant was briefed on the authentication system, calibrated with an eye tracker on a 1900×1200 px monitor, and was trained on the interface for a maximum of 2 minutes.

6.1. Dynamic Authentication Interface

We evaluated the dynamic interface under two conditions by varying the animation time. To evaluate the system accuracy in recognizing the true password and authenticating the user, each user entered a password that was selected randomly by the experiment facilitator. This procedure was repeated for a total of two true passwords. Next, to test the system's ability to reject the false password, the experiment facilitator set a different password (unknown to user), and the participant

attempted to access the system by guessing the password. This is similar to testing the system with true negatives, and this procedure was repeated for a total of two passwords. We tested the above procedures, i.e., entering two true and two false passwords, under two conditions by setting the animation speed of the interface to 3 seconds and 2 seconds. We tested two animation speeds since the goal was to achieve lower authentication time while supporting high accuracy.

6.1.1 System Accuracy

The system accuracy on the dynamic interface for 3 and 2 seconds animations are listed in Table 1.

Table 1: Dynamic interface - 3 & 2 Seconds Animations: Confusion Matrix [TP-True Password, FP-False Password], Acc-Accuracy, F-F_Measure (higher F[0 to 1] value is better)

	3 Seconds				2 Seconds			
	TP	FP	Acc	F	TP	FP	Acc	F
TP	85%	15%	92.5%	0.92	82.5%	17.5%	91.25%	0.90
FP		100%				100%		

6.1.2 Recognition Error

For a deeper analysis of the recognition accuracy, as shown in Table 2, we computed how many of the true passwords entered by the user were correctly recognized (0 error) by computing the Levenshtein distance [11]. Levenshtein distance is a measure of the number of entries in the password recognized by the system that are correct and in the right place when compared to the true password entered. For example, if the true password “pink-green-yellow-white" entered by the user is recognized as “pink-green-yellow-white" the Levenshtein distance is 0, i.e., the password has no errors (accepted). However, a password recognized as “pink-red-yellow-white" has a Levenshtein distance of 1 (rejected) since the system misrecognized “green" as “red."

Table 2: Dynamic Interface: recognition error based on the Levenshtein distance (higher 0 error is better).

Levenshtein distance	3 Seconds	2 Seconds
0 Error	**85%** (34/40)	**82.5** (33/40)
1 Error	**15%** (6/40)	**17.5%** (7/40)

6.1.3 Discussion

On the dynamic interface, sometimes, even when the participants followed the correct circle, the system wrongly recognized the circle followed. This kind of error is due to two circles having similar paths, and this is a limitation with the interface having many moving circles. Furthermore, the users could consistently keep track of the circles at the animation speed of 3 seconds, but found it a little difficult at 2 seconds resulting in reduced accuracy.

6.2. Static-Dynamic Authentication Interface

We followed the same evaluation procedure as the dynamic interface. Each participant entered two true and two false passwords under two conditions, i.e., the animation speed of the interface was set to 3 seconds and 2 seconds.

6.2.1 System Accuracy

The system accuracy on the static-dynamic interface for 3 and 2 seconds animations are listed in Table 3.

Table 3: Static-dynamic Interface - 3 & 2 Seconds Animation: Confusion Matrix [TP-True Password, FP-False Password], Acc-Accuracy, F-F_Measure (higher F is better).

	3 Seconds				2 Seconds			
	TP	FP	Acc	F	TP	FP	Acc	F
TP	97.5%	2.5%	98.75%	0.99	95%	5%	97.5%	0.97
FP		100%				100%		

6.2.2 Recognition Error

Table 4 show the recognition errors for all the true passwords entered under two experiment conditions (3 and 2 seconds animations) on the static-dynamic interface.

Table 4: Static-dynamic Interface: recognition error based on the Levenshtein distance (higher 0 error is better).

LV distance	3 Seconds	2 Seconds
0 Error	**97.5%** (39/40)	**95%** (38/40)
1 Error	**2.5%** (1/40)	**5%** (2/40)

6.2.3 Discussion

With the static-dynamic interface, participants expressed that it was easy and required less attention to follow the moving circles even when the animation speed was set to 2 seconds. This was due to fewer or no overlapping circles, since only 5 circles move during an animation. Also, to compare the accuracy, static-dynamic interface outperforms dynamic interface both with 3 (dynamic 92.5%, static-dynamic 98.75%) and 2 (dynamic 91.25%, static-dynamic 97.5%) seconds animations. Furthermore, there was no statistical difference

in the accuracies of the static-dynamic interface between 3 and 2 seconds (matched-pairs t-test P = 0.58 > 0.05). Since lower authentication time is better, all these factors strongly suggest that the static-dynamic interface with 2 seconds animation is the most practical solution.

7. Gaze- and PIN-based Authentication

To compare the accuracy of our authentication interface and its susceptibility to video analysis attacks through multiple threat models, we developed a Gaze- and PIN-based authentication system. The PIN-based authenticating interface uses a standard layout of numbers arranged in a 4×3 grid as seen on most ATMs. For consistency in comparison, the numeric grid was also placed in a the space of a 800×800 px square as shown in Figure 4b. A user authenticates with a PIN of 4 digits, and to enter the PIN with gaze, the user first looks at the digit and presses a hot key (A). We chose a hot key as we wanted to be consistent with the activation method on the colored circles interface where an animation is initiated by pressing a hot key(A).

7.1. PIN Recognition

PIN recognition is a simple process that uses Euclidean distance between the points. For each recorded gaze point, we compute the Euclidean distance to the center of every digit on the interface. The digit at the least distance from the gaze point is selected as the digit entered by the user. If all the 4 digits entered by the user match with the PIN, the user is authenticated.

7.2. Evaluation and Results

The same set of participants who evaluated gaze gesture-based interfaces also evaluated the gaze and PIN-based interface. Also, similar to the other interfaces, each participant entered two true and two false passwords.

7.2.1 System Accuracy

Table 5 and Table 6 show the system accuracy and recognition error respectively.

Table 5: PIN Interface: Confusion Matrix, Authentication Accuracy, and F-Measure (higher F value is better).

	True Pass	False Pass	Accuracy	F-Measure
True Pass	75%	25%	87.5%	0.86
False Pass		100%		

Table 6: PIN Interface: Recognition error based on the Levenshtein distance.

0 Error	1 Error	2 Error	4 Error
75% (30/40)	**17.5%** (7/40)	**2.5%** (1/40)	**5%** (2/40)

8. Threat Models

For hacking numeric password and colored circles password, we assume two threat models: 1) single video iterative attack, and 2) dual video iterative attack. Figure 5a shows the placement of front and back cameras used to record videos of the users while authenticating.

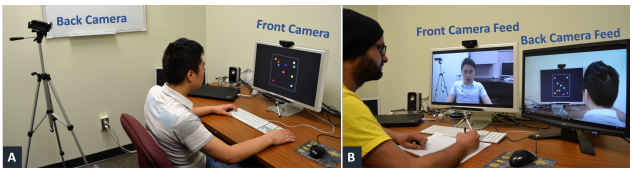


Figure 5: a) Authentication: front and back camera positions for both single and dual video threat models, b) Hacking: a user emulating a hacker is trying to guess the colored circles password through dual video iterative attack (static-dynamic interface).

8.0.1 Single Video Iterative Attack

In this model of attack, the attacker is made available a video stream (front camera) of the user's face showing clearly the eye movements while authenticating. This is similar to a casual observer focusing on the eyes of the victim while the victim is authenticating on a gaze-based system. Here, availability of the video extensively helps the attacker since the attacker can watch the video any number of times, and control the video playback for deeper analysis.

8.0.2 Dual Video Iterative Attack

In the dual video iterative attack, the attacker is made available two video streams: 1) a video stream (front camera) of the user's face showing clearly the eye movements while authenticating, 2) a video stream (back camera) of the authentication interface showing clearly any dynamic changes on the interface. For example, a video of the interface showing the transition paths of the circles, their colors, and the circles interchanging their positions for each new animation.

9. Password Hacking

We recruited 12 participants (9 male, 3 female) whose ages varied between 18 and 30 ($\mu_{age} = 23.75$). Each participant attempted to hack 2 numeric passwords and 2 colored circle passwords, and the order of passwords presented was counterbalanced. For each password, the participant was given a comfortable time limit of 10 minutes or 3 tries, whichever is earliest, as real-world systems (ATMs) block access after 3 unsuccessful attempts. Also, the participant was given complete control over the video playback as she could watch the video multiple times, pause, play, seek, etc.

9.1. Numeric Password Hacking

Numeric password hacking was evaluated under a threat model of single video iterative attack (video of the user's face), and this attack requires no second video since the interface does not change. Each participant was provided a video of a user authenticating using a numeric password. The video was randomly chosen from a set of videos recorded in the first phase.

Table 7: Numeric Password Hacking: the number of passwords recognized across each try.

Video	1st Try	2nd Try	3rd Try	Total
Single	4 (16.7%)	6 (25%)	9 (37.5%)	79.2%

A total of 24 passwords were attacked (12 x 2), and 79.2% (19/24) passwords were correctly recognized. The cumulative time taken to attack all the 24 passwords was 104 minutes, leading to an average time of 4.33 minutes spent on hacking a password either successfully or not. Table 7 shows the number of passwords recognized across each try.

9.2. Colored Circles Password Hacking

For hacking the authentication based on moving colored circles strategy, we considered the static-dynamic interface (800×800 dimension and 2 seconds animation) as it was found to be the most practical authentication method compared to dynamic interface. Hacking passwords on the static-dynamic interface was evaluated under two threat models.

Table 8: Colored Circles Password Hacking: number of passwords recognized across each try.

Video	1st Try	2nd Try	3rd Try	Total
Single	0	0	0	0%
Dual	4 (16.7%)	0	0	16.7%

9.2.1 Single Video Iterative Attack

Each participant was provided the video (front camera - user's face) of a user authenticating on the static-dynamic interface (chosen randomly). A total of 24 passwords were attacked (12 x 2), and 0% (0/24) passwords were correctly recognized (Table 8). The cumulative time taken to attack all the 24 passwords was 58 minutes, leading to an average time of 2.42 minutes spent on hacking a password either successfully or not. As expected, no password was hacked since the participant only had access to the video stream showing the user's face, but the interface was dynamic, and the circles move and change their positions randomly.

9.2.2 Dual Video Iterative Attack

This is an advanced attack, as we are assuming that the hacker has access to two videos, one showing the user's face (front camera) and the other showing the authentication interface (back camera) as shown in Figure 5b. Each participant was provided with two videos (user's face and interface) of a user authenticating (chosen randomly). A total of 24 passwords were attacked (12 x 2), and 16.7% (4/24) passwords were correctly recognized. The cumulative time taken to attack all the 24 passwords was 202 minutes, leading to an average time of 8.46 minutes spent on hacking a password either successfully or not. While this kind of attack is sophisticated, our system is still resilient to such attacks. Table 8 shows the number of passwords recognized across each try.

We hypothesized that participants would not be able to recognize the password even with dual videos, and even if they do, it would take more than 3 tries. Interestingly, 4 passwords were recognized in the first try. 1 participant recognized 2 passwords, and 2 participants recognized 1 each. During the interview, these participants shared that they could recognize the password as they were able to precisely sync both videos and identify the start and end of each animation. The video sync was achieved as the victim (user authenticating) was unknowingly giving out the information about the start and end of the animation by either hitting the hot key hard (animation trigger) or was taking long pauses between animations.

10. Discussion

We discuss how the accuracy and resilience to video analysis attacks are influenced by various parameters of the system.

Interface Dynamics Vs Accuracy: By comparing the accuracies and feedback from the users for both the dynamic and static-dynamic interfaces, it is suggestive that as the interface becomes more dynamic (all moving circles) the users find it overwhelming for a focused task like authentication, resulting in reduced accuracy. In addition, to achieve high accuracy when using gaze gesture-based authentication, the interface should support a moderate margin of error. For example, on the static-dynamic interface only 5 circles move during an animation, and this reduces the chances of generating similar paths–i.e., supporting a moderate margin of error–unlike the dynamic interface.

Authentication Time Vs Accuracy: Considering only the static-dynamic interface, it can be observed that for animation speed of 2 seconds or higher the accuracy does not differ much (3 seconds 98.75%, 2 seconds 97.5%, $P > 0.05$). Hence, with an animation speed of 2 seconds, considering a 4 colors password, the least authentication time is 8 seconds. As we found through additional studies,

reducing the authentication time by reducing the animation speed to 1 second (circles move fast) brings down the accuracy to 70%. However, with 2 seconds animation, based on the level of security required, reducing the password length to less than 4 colors will reduce the authentication time to below 8 seconds.

Interface Dynamics Vs Video Analysis Attacks: From the "Password Hacking" study we found that our interface with moving circles was more secure than the static (PIN) interface. 79.2% of the passwords were recognized on the PIN (static) interface. The moving circles interface was not susceptible to single video iterative attack, and had a low success rate with dual video iterative attacks (16.7%). We believe that future gaze-based authentication systems should adopt an interface with dynamic events to prevent video analysis attacks. However, it needs to be ensured that such a dynamic interface does not overwhelm the user.

Limitations: Few limitations of our approach are that users with colorblindness will have limited set of colors to choose from as they can not distinguish few colors. A solution would be to have different shapes with different colors, while creating no visual overload. Furthermore, an authentication time of ~ 8 seconds would not be appropriate in cases where the authentication is performed frequently.

11. Conclusion and Future Work

We presented a dynamic gaze gesture-based authentication system to counter shoulder surfing attacks. We also explored two authentication interfaces: 1) a dynamic interface, and 2) a static-dynamic interface that leverage gaze gestures. Through system evaluations, we found that the static-dynamic interface with an animation speed of 2 seconds is the most practical solution of the two interfaces. We further evaluated the resiliency of our authentication method to video analysis attacks, and found that our system is not susceptible to single video iterative attacks, and has lower success rate with dual video iterative attacks compared to a PIN- and gaze-based system. As future enhancements, we will try to reduce the authentication time and improve the password memorability through interface modifications.

References

[1] D. S. Best and A. T. Duchowski. A rotary dial for gaze-based pin entry. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, ETRA '16, pages 69–76. ACM, 2016.

[2] A. Bulling, F. Alt, and A. Schmidt. Increasing the security of gaze-based cued-recall graphical passwords using saliency masks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12. ACM, 2012.

[3] D. H. Cymek, A. C. Venjakob, S. Ruff, O. H.-M. Lutz, S. Hofmann, and M. Roetting. Entering pin codes by smooth pursuit eye movements. *Journal of Eye Movement Research*, 2014.

[4] A. De Luca, M. Denzel, and H. Hussmann. Look into my eyes!: Can you guess my password? In *Proceedings of the 5th Symposium on Usable Privacy and Security*, SOUPS '09, pages 7:1–7:12. ACM, 2009.

[5] A. De Luca, R. Weiss, and H. Drewes. Evaluation of eye-gaze interaction methods for security enhanced pin-entry. In *Proceedings of the 19th Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces*, OZCHI '07, pages 199–202. ACM, 2007.

[6] M. Eiband, M. Khamis, E. von Zezschwitz, H. Hussmann, and F. Alt. Understanding shoulder surfing in the wild: Stories from users and observers. In *Proceedings of the 35th Annual ACM Conference on Human Factors in Computing Systems*, CHI '17. ACM, 2017.

[7] P. Institute. Global Visual Hacking Experimental Study: Analysis. 2016.

[8] M. Khamis, F. Alt, M. Hassib, E. von Zezschwitz, R. Hasholzner, and A. Bulling. Gazetouchpass: Multimodal authentication using gaze and touch on mobile devices. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 2156–2164. ACM, 2016.

[9] M. Kumar, T. Garfinkel, D. Boneh, and T. Winograd. Reducing shoulder-surfing by using gaze-based password entry. In *Proceedings of the 3rd Symposium on Usable Privacy and Security*, SOUPS '07, pages 13–19. ACM, 2007.

[10] A. Leon-Garcia and A. Leon-Garcia. *Probability, statistics, and random processes for electrical engineering*. Pearson/Prentice Hall 3rd ed. Upper Saddle River, NJ, 2008.

[11] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[12] J. Long. *No tech hacking: A guide to social engineering, dumpster diving, and shoulder surfing*. Syngress, 2011.

[13] V. Rajanna and T. Hammond. Gawschi: Gaze-augmented, wearable-supplemented computer-human interaction. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, ETRA '16, pages 233–236, New York, NY, USA, 2016. ACM.

[14] V. Rajanna, P. Taele, S. Polsley, and T. Hammond. A gaze gesture-based user authentication system to counter shoulder-surfing attacks. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17. ACM, 2017.

[15] M. Vidal, A. Bulling, and H. Gellersen. Pursuits: Spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13. ACM, 2013.

[16] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 159–168. ACM, 2007.