Gaze-Based Password Authentication through Automatic Clustering of Gaze Points

Justin Weaver, Kenrick Mock
Computer Science
University of Alaska Anchorage
Anchorage, AK, USA
jiweaver, kenrick@uaa.alaska.edu

Bogdan Hoanca
Computer Information Systems
University of Alaska Anchorage
Anchorage, AK, USA
afbh@uaa.alaska.edu

***Abstract*— Researchers have proposed systems in which users utilize an eye tracker to enter passwords by merely looking at the proper symbols on the computer monitor in the appropriate order. This authentication method is immune to the practice of shoulder surfing: secretly observing the keystrokes of a legitimate user as he or she types a password on a keyboard. In this paper we describe the EyeDent system—in which users authenticate by looking at the symbols on an on-screen keyboard to enter their password. Existing eye-tracking based authentication systems require the user to dwell or press a trigger when looking at each symbol. Instead, in EyeDent, gaze points are automatically clustered to determine the user's selected symbols; this approach has the benefit of allowing users to authenticate at their natural speed, rather than with a fixed dwell time. Additionally, the absence of a visible trigger does not divulge the number of symbols in the password. Results from preliminary investigations indicate that quick (3 seconds for a 4 digit PIN) authentication is possible using this scheme, but more work is needed to account for calibration error, and to dynamically adapt system parameters to the characteristics of individual users.**

Keywords- eye tracking, authentication, gaze-based

I. Introduction

Flawless identity management remains a critical and intractable problem, and is consequently the focus of many intensive research efforts. Current security techniques are weak in general, because most are easily circumvented or fooled. For example, a password can be compromised, cracked, or even just forgotten. A hardware key can likewise be lost, duplicated, or stolen.

Shoulder surfing is the name given to the practice of observing a legitimate user as he or she authenticates, to impersonate the legitimate user later. Shoulder surfing can be done in-person, or can also potentially be accomplished using a simple, properly-positioned video camera.

The nefarious threat of shoulder surfing can be virtually eliminated by literally taking the login out of the user's hands. When the user enters their password merely by looking at the appropriate symbols on the screen, instead of typing the symbols on a keypad, shoulder surfing becomes practically impossible. This advanced authentication technique is possible to achieve using an eye tracker.

In this project we developed the EyeDent system which presents an on-screen keyboard (or keypad) to the user, and allows the user to authenticate by looking at the symbols in his or her password in order. By using an on-screen keyboard we maintain compatibility with traditional password schemes, which may still be used in secure settings such as one's home or office. The user-selected symbols are determined using an automatic clustering algorithm (described in Section III).

We implemented EyeDent using an EyeTech Digital Systems TM3 eye tracker [1]; this is a remote eye tracker accurate to 0.5 degrees that tolerates head motion within a 25 x 16 x 19 cm window. Prior to developing EyeDent, we created an API wrapper in the form of a DLL that allows programs written in .NET languages such as C# to interface with the eye tracker using EyeTech's QuickLink API. The source code is publicly available under the MIT open source license at http://code.google.com/p/quicklinkapi4net/.

II. Related Work

Many authentication approaches attempt to thwart shoulder surfing by obfuscating the shared secret between the user and the authenticating system with some random element [2]. For example, in the ColorLogin scheme, a user password consists of a set of icons known only by the user and the authenticating system. Lines of icons are randomly displayed with an icon selected from the user's password set and icons not in the user's password set. The user clicks only on the lines that contain icons in his or her password set. A click only reveals that one of the icons in the line is a member of the user's password set, but does not reveal which icon was recognized. Subsequent logins present a different set of generated icons to thwart a shoulder-surfer [3]. Such randomized schemes, in which users perform some mental computation, have been proposed by many researchers [4,5]; however, they are vulnerable to attack, if an attacker can record multiple authentication sessions. In particular, an intersection attack involves looking for those icons that appear in repeated sessions; the attacker can eliminate icons that only appear in some of the successful login sessions, as those cannot be the icons in the user's password [6].

Graphical implementations of passwords resistant to shoulder surfing have also been proposed. In Sobrado and Birget's scheme, the shared secret consists of a set of icons randomly distributed on the screen [7,8,9]. The user authenticates by clicking anywhere inside the convex hull define d by the icons in the user's password. The attacker cannot easily find out which icons define the hull, because there are multiple possibilities for any given click point. The scheme is easy to understand in principle, but has several weaknesses. First, locating the icons on the screen can be difficult, because several icons may look similar. Second, the authentication process tends to be lengthy to avoid false positives, where a random guess matches the required entry. Thirdly, even with a lengthy authentication process, the attacker has a relatively high chance of authenticating by randomly guessing at click points. Finally, the scheme is still vulnerable to repeat observations and intersection attack.

More recently, eye tracking technologies have been used for password entry to protect users from shoulder surfing attacks under the assumption that it is difficult for an attacker to correlate eye movements to symbols on the screen. Hoanca and Mock proposed utilizing an eye tracker in combination with a graphical password [10]. Simpler techniques use eye gestures as passwords that decrease the authentication time, but at the cost of a relatively small password space (eight input strokes–or symbols), and still require users to recall and perform special tasks to authenticate [11].

In Kumar *et al.*'s approach, a user's gaze is tracked while looking at an on-screen keyboard. Successive symbols in the password are entered by dwelling on an on screen key, or by pressing a trigger key with a finger while looking at a symbol on the screen [12]. Kumar *et al.* found that the gaze + trigger method suffers from high error rates due to variance in eye-hand coordination: some users press the trigger before looking at the target area or after leaving the target area. To address this problem, researchers have implemented trigger correction algorithms, inserted focus points to attract gaze [13], and collected multiple gaze points while holding down the trigger key to calculate a gaze centroid [14,15].

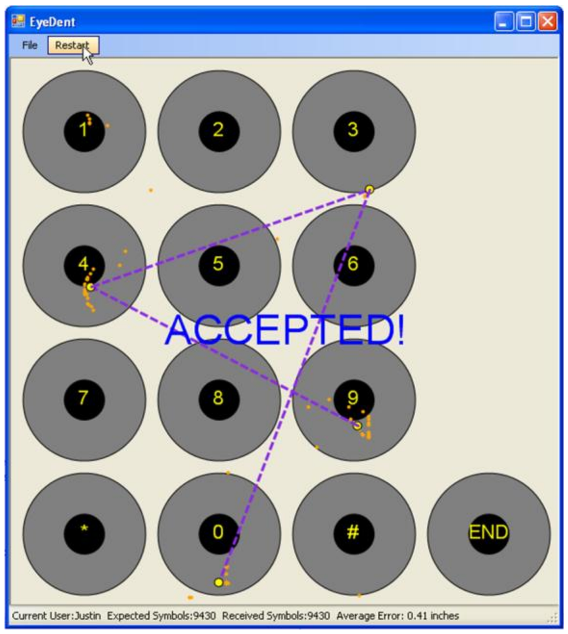
Both Kumar *et al.* [12] and Forget *et al.* [14] focused their efforts on dwell-time algorithms that require the user to consciously pause the movement of their eyes and continue to gaze at each symbol for a predetermined amount of time. With the dwell based schemes, a shorter dwell time leads to less accurate authentication for those users that would naturally prefer a longer dwell time. The matching approaches employed in this project overcome this limitation by analyzing the user's gaze pattern in a way that is more adaptive to the user's natural gaze speed. Finally, our scheme does not disclose the number of symbols in the password and the associated entry timing, as in Kumar *et al.*'s method, which requires a visually observable trigger, or an audio tone to cue users that a symbol had been entered.

III. EYEDENT OVERVIEW AND DESIGN

EyeDent presents a simple GUI with an on-screen keyboard in which the buttons are sized so the keyboard's window fits within the width of the screen. Each button on the keyboard is a large circle with a symbol in the center (Fig. 1). The intent of this design is to draw the user's gaze to the center of the circle representing each symbol [13].

EyeDent begins with a calibration step. While we used a somewhat time-consuming 16-point calibration, a 1-step calibration or other faster version is also possible [14]. Once calibrated, the eye tracker settings are stored per user, and re-calibration is not necessary, unless the eye tracker has moved with respect to the screen. Next, as the user looks at the password symbols, EyeDent takes a constant stream of data from the eye tracker (30 samples per second), and places each data sample on a queue. Each sample consists of the user's gaze coordinates along with other information about their eye (see Section III-D, Data Logging and Average Error Analysis).

To authenticate, a user must look briefly at each symbol in his or her password, without interruption, and in sequence; and then look at the END button to signal the completion of the process. As each user has a longer or shorter dwell time on successive symbols, this protocol allows the user to follow their natural dwell time preferences, and also to look at different symbols (not in their password) without penalty, as long as the last set of symbols viewed matches the user's password.



Current User:Justin Expected Symbols:9430 Received Symbols:9430 Average Error: 0.41 inches

Figure 1. A successful login on a numeric-style keypad with the password "9430". The small dots are individual gaze points. The slightly larger circles are clusters. The dashed lines aid our visual inspection by connecting the clusters to make the user's gaze path obvious. The dots and lines are not shown during actual authentication. Notice that some gaze points are recorded near symbol "1", but too few points are recorded to constitute a cluster.

Once the END button has been triggered, EyeDent stops taking new data, and processes the current queue of samples. First, it partitions the data samples into clusters (see Sections III-A and III-B, Cluster Partitioning). Next, it performs the password matching by using the location and temporal order of the clusters. A cluster within a symbol's circle triggers that symbol (see Password Matching). EyeDent will then indicate whether the password was accepted or rejected.

We compared the performance of two different clustering algorithms: one based on a fixed minimum number of points per cluster (Section III-A), and one based on a dynamic minimum number of points per cluster (Section III-B).

A. *Clustering – Fixed Minimum Points per Cluster*

The first step in the password matching is partitioning the data points in the input queue. Groups of temporally continuous data points are lumped into clusters composed of points that lie close together.

Our initial algorithm to aggregate the data points requires a minimum number of nearby gaze points to constitute a cluster. We begin with an empty cluster C_x, and then add the first data point P_n. Then we examine the next data point P_{n+1}. If P_{n+1} is within a specified fixed threshold distance of the centroid of C_x, then P_{n+1} is added to C_x, and we proceed to the next data point P_{n+2}; otherwise C_x is sealed and placed on a special queue, and then a new empty cluster C_{x+1} is created with initial point P_{n+1}. We proceed as we did with C_x, by adding points to C_{x+1}, and then C_{x+2}, and so on until all the data points are partitioned into clusters. The process is depicted in Fig. 2.

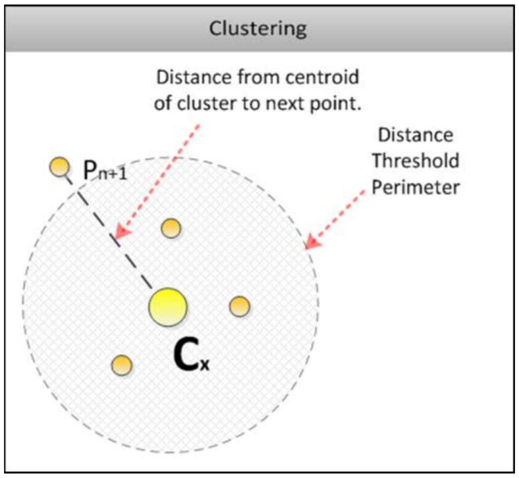

Clustering
Distance from centroid of cluster to next point.
Distance Threshold Perimeter
Pn+1
Cx

Figure 2. The clustering algorithm in action. The point P_{n+1} lies outside the distance threshold for the cluster C_x. The algorithm would seal cluster C_x, and make a new cluster C_{x+1}, with P_{n+1} as its initial point.

EyeDent allows configuration of the Minimum Points per Cluster (MPPC) parameter used during partitioning. Clusters with fewer than MPPC points are not considered valid dwell points and are discarded. By default, we set MPPC to 9. At a rate of 30 samples per second, this implies a minimum dwell time of 0.3 seconds per cluster. The Distance Threshold is also a configurable parameter that we set to the default value of 0.75 inches. To calculate the distance in inches on the monitor we explicitly enter the width of the screen in inches, which allows us to calculate pixels/inch based on the resolution setting.

B. *Clustering – Dynamic Minimum Points per Cluster*

We also experimented with an algorithm to dynamically calculate the Minimum Points per Cluster parameter. The advantage of dynamically computing this parameter is that dwell time varies by user. A fast user could look at a symbol for fewer than 0.3 seconds, which would be missed by the algorithm in III-A. A slower user might dwell much longer, in which case unintended dwelling on non-password symbols would result in those symbols being counted in the user's password entry.

Our initial analysis suggested that the cluster size typically follows a bimodal distribution with either small or large clusters. The larger clusters generally correspond to the password symbols, while the small ones result from distractions or unintended dwell on non-password symbols. We used a hierarchical agglomerative clustering algorithm to find a threshold between the two peaks of the distribution.

In this algorithm, cluster S is initially set to the smallest cluster, and cluster L is initially set to the largest cluster. The algorithm iterates through every other cluster C_i, and calculates $|\text{Size}(C_i) – \text{AverageSize}(S)|$ and $|\text{Size}(C_i) – \text{AverageSize}(L)|$. C_i is then added into S or L respectively, whichever is closer. The Minimum Points per Cluster parameter was then set to $(\text{AverageSize}(S) + \text{AverageSize}(L)) / 2$.

C. *Password Matching*

The actual password matching algorithm keeps two pointers as it operates: one pointer into the list of clusters C_x, and another pointer into the list of expected password symbols S_n. In each case, the pointers start at the end of the lists and progress toward the beginning, i.e., backwards. If a cluster point C_x is located over a keypad button B, then the symbol for that button is compared to the next expected password symbol S_n. If the symbol of button B is equal to S_n, then the cluster C_x is assigned to S_n, and we proceed to the next cluster C_{x-1}. Otherwise, we continue with the next expected password symbol S_{n-1}. However, first we check the number of clusters assigned to symbol S_n. If S_n contains 0 clusters, then the match fails immediately. If the algorithm reaches the first symbol in the expected password (S_0) without failing, and at least one cluster occurs on the symbol S_0, the match is successful.

Note that, if a cluster point C_x falls into the dead-zone between buttons, it is ignored (with one notable exception), and the algorithm proceeds to examine C_{x-1}. The exception is that those dead-zone clusters are recognized as separators between multiple continuous occurrences of the same symbol within a password, e.g., the double "o" in the word "poodle". This algorithm treats temporally contiguous clusters that translate into the same symbol as a single occurrence of that symbol.

D. *Data Logging and Average Error Analysis*

During authentication, the average error is computed by performing a best-fit analysis of the clusters and password symbols. The algorithm works in much the same fashion as the password matching algorithm, i.e., it matches clusters to password symbols (from back to front). However, as the algorithm progresses, clusters are assigned to the user's password symbols based on which symbol they are closest to, rather than which button they are on. Then, the algorithm calculates the average distance (in inches) between the center of each symbol and the clusters we assigned to it. Finally, the results of all of those calculations are averaged, and the final value is reported

Although the Average Error value is a flawed metric (on its own), since it does not describe the specifics of the distribution, it does provides an approximate idea of just how far a login attempt was from being successful. If the average error displayed is NaN (Not a Number), it means the attempt was not even close, because at least one symbol of the expected password was left without any corresponding clusters.

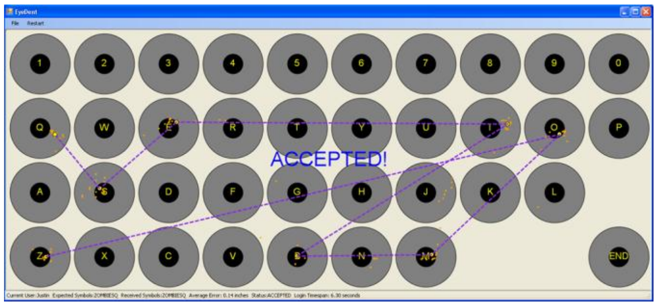
To further aid in data analysis, every authentication attempt generates two log files. The main log file contains the raw data from the eye tracker (the gaze point samples), in CSV text format. This data consists of flags indicating if the eyes are found and calibrated, coordinates of the glint points in the camera image (the reflections from the tracker's infrared lights), pupil diameter, and gaze coordinates on the screen. The secondary log file contains data about the analysis of the login attempt, and configuration information for the authentication session.

IV. Preliminary Investigation

We have only conducted preliminary work to evaluate the algorithm. In this section, we present results on repeated authentication attempts by the three authors on an alphanumeric keyboard layout. The experiments were run on a desktop computer with a 19-inch screen width. EyeDent was configured to display 1.3-inch diameter buttons, with 0.125-inches of padding between each button. Calibration was performed at the beginning of each session.

We used a QWERTY layout for users to enter the password of "ZOMBIESQ". We specifically chose this password because it requires looking near all four corners of the virtual keyboard (where the eye tracker error is typically highest), includes symbols from all the letter rows, and has three symbols close to each other (i.e., "Z", "S", and "Q"). Fig.3 shows a screen capture of the layout for a successful authentication attempt. We also tested a numeric keypad layout with the password "9430".

Our initial study used a fixed MPPC of 7, which resulted in successful authentication rates ranging from 35-75% by the three authors. Many of the unsuccessful attempts were due to



Current User: Justin Expected Symbols: ZOMBIESQ Received Symbols: ZOMBIESQ Average Error: 0.14 inches Status: ACCEPTED Login Timespan: 6.30 seconds

Figure 3. Alphanumeric keyboard layout showing a successful login attempt for the password "ZOMBIESQ".

extra inserted characters in the entered password. We calculated the average cluster size for the extra characters to be 8.25, which was just barely above our chosen threshold of 7; this directly suggested that we could use a larger threshold, such as 9, to eliminate these errors. Furthermore, the larger cluster size should not influence performance, since the shortest average dwell time for successful authentications was over 500ms, which corresponds to a cluster size of about 16 points.

A. *Alphanumeric Password Entry, 9 MPPC*

Table I summarizes results for the scheme using a fixed Minimum Points per Cluster of 9. The table displays the percentage of successful authentications out of *N* attempts, average total authentication time for the accepted attempts, average dwell time per symbol for accepted attempts, and average error per symbol (distance of cluster centroid from the center of the button).

TABLE I. "ZOMBIESQ" Accepted Authentication Results

	Success	**Ave Total Time**	**Ave Dwell Time**	**Ave Error**
User 1 (N=12)	83%	7.3s	820ms	0.26"
User 2 (N=12)	83%	7.9s	745ms	0.31"
User 3 (N=12)	83%	5.8s	612ms	0.37"

Table II provides further detail about the rejected authentication results. Out of *R* rejected attempts, the percentage of rejected cases is broken down by single error or multiple errors. The errors are categorized by cases **M**, **W**, and **E**. Case **M** refers to the situation where a character is missing from the password. For example, given the password of "ZOMBIESQ", the entry of "ZOMBIEQ" (missing an "S") is a case of a missing character. Case **W** refers to the situation where a wrong character was substituted for a correct character; for "ZOMBIESQ", the entry of "ZOMBUESQ" ("U" is substituted for "I") is an example of this case. Finally, Case **E** refers to the situation where an extra character is inserted into the password; for "ZOMBIESQ", the entry of "ZOMNBIESQ" (an extra "N") is an example of this case.

TABLE II. "ZOMBIESQ" Rejected Authentication Results. **M**=Missing Character, **W**=Wrong Character Substituted, **E**=Extra Character, **Combo**=Combination of **M**,**W**, or **E**. Numbers in each row add up to 100% (aside from rounding errors)

	Single Error			**Multiple Errors**			
	M	**W**	**E**	**M**	**W**	**E**	**Combo**
User 1 (R=2)	50%		50%				
User 2 (R=2)						100%	
User 3 (R=2)	50%		50%				

The results from Table II consisted of consecutive authentication attempts performed in one session. Under these circumstances, it seemed that establishing a visual rhythm based on short-term muscle memory might affect the results. To investigate long-term vs. short-term muscle memory, the investigators also performed a single daily authentication over five days. User 1 successfully authenticated on the first attempt on four out of five days; the remaining attempt succeeded on the second try. User 3 successfully authenticated on the first attempt on three out of five days; the remaining two attempts were successful on the second try.

B. Dynamically Calculated Minimum Points per Cluster

The authors also attempted to authenticate using the password of "ZOMBIESQ", but using the dynamic algorithm described in Section III-B to calculate the Minimum Points per Cluster (MPPC). Results are given in Tables III through V. The authentication time varies since several users intentionally dwelled on symbols for a very short time or for a very long time to see if the algorithm would generate an appropriate threshold.

TABLE III. DYNAMIC MINIMUM POINTS PER CLUSTER - "ZOMBIESQ" ACCEPTED AUTHENTICATION RESULTS

	Success	**Range - Total Time**	**Ave Error**
User 1 (N=15)	47%	3.9-13.5s	0.38"
User 2 (N=9)	44%	7.4-10.9s	0.44"
User 3 (N=12)	42%	5.1-17.0s	0.42"

TABLE IV. DYNAMIC MINIMUM POINTS PER CLUSTER - "ZOMBIESQ" REJECTED AUTHENTICATION RESULTS. **M**=MISSING CHARACTER, **W**=WRONG CHARACTER SUBSTITUTED, **E**=EXTRA CHARACTER, **COMBO**=COMBINATION OF **M**,**W**, OR **E**. NUMBERS IN EACH ROW ADD UP TO 100% (ASIDE FROM ROUNDING ERRORS)

	Single Error			**Multiple Errors**			
	M	**W**	**E**	**M**	**W**	**E**	**Combo**
User 1 (R=8)	38%	25%	25%				12%
User 2 (R=5)	80%			20%			
User 3 (R=7)	14%			43%	29%		14%

C. Numeric Password Entry, Four-Digit PIN, 9 MPPC

We also tested used the password of "9430" on a numeric keypad layout (Fig. 1) to simulate entering a PIN at an ATM. Users 2 and 3 both had two single errors under case **W**.

TABLE V. "9430" ACCEPTED AUTHENTICATION RESULTS

	Success	**Ave Total Time**	**Ave Dwell Time**	**Ave Error**
User 1 (N=12)	100%	2.7s	652ms	0.32"
User 2 (N=12)	83%	2.7s	557ms	0.50"
User 3 (N=12)	83%	2.7s	627ms	0.31"

V. ANALYSIS AND DISCUSSION

A. Nine Minimum Points per Cluster

Using the fixed MPPC of 9 each user achieved an authentication rate of 83% (see Table I). Further improvements are possible. First, a majority of the rejected authentications are due to single errors – a single character missing, inserted, or substituted. If the authentication process were relaxed to allow a single error then the successful authentication rate would remain unchanged for user 2 but would rise to 100% for users 1 and 3 (see Table II), but at the expense of a smaller password space.

There were no errors where a single incorrect character was substituted in place of a correct password character (see Table II – case **W**). This type of error would be expected if the eye tracker was poorly calibrated and a nearby symbol was erroneously entered instead of the actual gaze symbol. The errors were case **M** where a character was missing from the password, or case **E** where an extra character was inserted into the password. The case of the missing character suggests the user did not dwell on a symbol long enough to create a cluster – a problem that could potentially be addressed with a smaller MPPC. The case of the extra character suggests a spurious cluster that was registered as the user searched or scanned the keys – a problem that could potentially be addressed with a larger MPPC. None of the authentication attempts included both missing and extra characters. These errors could not be addressed by adjusting the MPPC.

User 1 experienced the lowest average error (see Table I) – defined as the average distance of the cluster's centroids from the actual center of their associated buttons. This may be due to environmental circumstances (e.g., better camera focus) or user behavior (e.g., head remained more motionless resulting in better accuracy). However, each user's error was within the cluster distance threshold of 0.75 inches.

Finally, the authentication rate appears similar when authenticating once daily compared to authenticating sequentially in the same session. This implies that short-term muscle memory controlling the eye does not play a significant factor in authentication success.

B. Dynamically Calculated Minimum Points per Cluster

The scheme to dynamically calculate the MPPC has the potential to alleviate some of the errors encountered by the static scheme. However, as shown in Table III and Table IV, the algorithm needs improvement, because the success rate was much lower. However, if single errors are allowed the success rate increases to 93%, 89%, and 50% respectively. User 3 experienced many errors due to a MPPC value that was often too large; as a result, multiple valid clusters were discarded.

Despite the errors, there are some promising results where the algorithm performed as expected. The total authentication time in Table III is given as a range, because users tried authenticating both quickly and slowly. For example, in Table III, user 1 authenticated as quickly as 3.9 seconds, and as slowly as 13.5 seconds. The 3.9-second authentication had a MPPC of 5, and is approaching the authentication speed of a keyboard. During the slow authentication, the user was able to look at non-password symbols quickly, but they were discarded due to the longer dwell time on the password symbols.

A more sophisticated analysis of the cluster size histogram should lead to a better MPPC, and thus improve the authentication results. For example, rather than assume a bimodal distribution, we might check for a normal distribution and if found then set the MPPC to a value below the median.

Another possibility is to use the cluster size of correctly matched password symbols to help determine an appropriate MPPC. For example, if the first symbol in the password is “Z”, and the log begins with 15 samples in the vicinity of the symbol “Z”, then 15 might factor into the MPPC threshold calculation for the other symbols.

C. *Numeric Password Entry, 9 MPPC*

As expected, authentication was more successful and quicker using a short 4-digit PIN on the numeric keypad layout than the longer password on the QWERTY layout. Three of the four errors resulted from dwelling on a blank area outside the keypad symbols – a more likely situation than the QWERTY layout due to the fewer number of symbols. In this case, we could likely eliminate some of these errors by mapping clusters outside the keypad to the closest symbol.

VI. Conclusions

The goal of EyeDent is to authenticate users via eye tracking without the need for special triggers or predefined dwell times. Our initial results suggest that this goal can be achieved. Using a fixed Minimum Points per Cluster of 9 resulted in mostly successful authentication attempts by the authors. More work needs to be done to determine if this success translates to general users and to account for calibration error, accuracy, and variation in user dwell times. Our results indicate that dynamically determining the Minimum Points per Cluster does support user variation in dwell times, but more work is required to adjust the algorithm to increase the authentication rate. A dynamic algorithm would also improve portability to other eye trackers with a different sampling rate. A related topic for consideration is a more optimal clustering algorithm than the greedy in-order clustering algorithm shown in Fig. 2.

Other considerations include an allowance for single errors or probabilistic acceptance based on distance from the target symbol instead of a discrete match of clusters to symbols. For example, if the mean squared error of all authentication points is within a threshold, then the algorithm might deem the attempt as successful. This could allow a cluster to be far from its target if all other clusters are close to their target.

Another item for future work is to compute the MPPC from our log data that maximizes success and minimizes error and see how it compares to the selected value of 9.

Finally, in this project we performed a separate calibration step before performing authenticating. One approach to integrate calibration into the authentication process is to highlight different buttons on the virtual keyboard. This would also have the benefit of calibrating specifically for the keyboard being displayed while familiarizing users with the layout. It may also be possible to eliminate calibration, if the raw glint data from the eyes can be proportionally mapped to the same vector motions that the eyes would make when gazing at symbols in the password.

Acknowledgment

We would like to acknowledge Liz Weaver and Kenneth Kelley for their assistance testing EyeDent, and the University of Alaska Anchorage’s College of Arts & Sciences and the University Honors College for financial support.

References

[1] EyeTech Digital Systems. Retrieved February 23, 2011 from http://www.eyetechds.com/

[2] Tari, F., Ozok, A. A., and Holden, S. H. A comparison of perceived and real shoulder surfing risks between alphanumeric and graphical passwords. In Proceedings of the Second Symposium on Usable Privacy and Security (Pittsburgh, Pennsylvania, July 12 - 14, 2006). SOUPS '06, vol. 149. ACM Press, New York, NY, 2006, 56-66.

[3] Haichang Gao, Xuewu Guo, Xiaoping Chen, Liming Wang, and Xiyang Liu. YAGP: Yet Another Graphical Password Strategy. In Proceedings of the 2008 Annual Computer Security Applications Conference (ACSAC '08). IEEE Computer Society, Washington, DC, USA, 2008, 121-129.

[4] Matsumoto, T. Human-computer cryptography: An attempt, 3rd ACM Conference on Computer and Communications Security, pp. 68-75, New Delhi, March 1996.

[5] Hopper, N. and Blum, M. A Secure Human-Computer Authentication Scheme, CMU Tech Report CMU-CS-00-139, 2000. Retrieved January 25, 2008, from http://reports-archive.adm.cs.cmu.edu/anon/2000/CMU-CS-00-139.pdf

[6] Hoanca, B. and Mock, K. A Theoretical Framework for Assessing Eavesdropping-Resistant Authentication Interfaces. The 2009 Hawaii International Conference on System Sciences, Waikoloa, HI, Jan. 5-8, 2009.

[7] Sobrado, L. and Birget, J.-C. Shoulder surfing resistant graphical passwords. 2005. Retrieved January 27, 2008, from http://clam.rutgers.edu/~birget/grPssw/srgp.pdf

[8] Sobrado, L. and Birget, J.-C. Graphical passwords, The Rutgers Scholar, vol 4, 2002. Retrieved January 25, 2008 at http://rutgersscholar.rutgers.edu/volume04/sobrbirg/sobrbirg.htm

[9] Wiedenbeck, S., Waters, J., Sobrado, L., and Birget, J. Design and evaluation of a shoulder surfing resistant graphical password scheme. In Proceedings of the Working Conference on Advanced Visual interfaces (Venezia, Italy, May 23 - 26, 2006). AVI '06. ACM Press, New York, NY, 177-184.

[10] Hoanca, B. and Mock, K.. Secure Graphical Password System for High Traffic Public Areas. In Proceedings of ETRA - Eye Tracking Research and Applications Symposium. San Diego, California, USA: ACM Press. pp. 35, 2006.

[11] De Luca, A., Denzel, M., and Hussman, H. Look into my eyes!: can you guess my password? SOUPS '09: Proceedings of the 5th Symposium on Usable Privacy and Security, 2009.

[12] Manu Kumar, Tal Garfinkel, Dan Boneh, and Terry Winograd. Reducing shoulder surfing by using gaze-based password entry. In Proceedings of the 3rd symposium on Usable privacy and security (SOUPS 2007). ACM, New York, NY, USA, 13-19

[13] Kumar, M., Klingner, J., Puranik, R., Winograd, T., and Paepcke, A. Improving the accuracy of gaze input for interaction. Proceedings of the 2008 symposium on Eye tracking research and applications (pp. 65-68). New York: ACM.

[14] Forget, A., Chiasson, S., & Biddle, R. Input precision for gaze-based graphical passwords. CHI EA '10: Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems.

[15] Forget, A., Chiasson, S., and Biddle, R. Shoulder surfing resistance with eye-gaze entry in cued-recall graphical passwords. CHI EA '10: Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems.