Password entry scheme resistant to eavesdropping

Bogdan Hoanca[1] **and Kenrick Mock**[2]
[1]Computer Information Systems, University of Alaska Anchorage, Anchorage, AK, USA
[2]Mathematical Sciences, University of Alaska Anchorage, Anchorage, AK, USA

Abstract - *We propose an authentication scheme resistant to eavesdropping attacks. Users select an alphanumeric password with a length of 9-15 symbols. They can use this password in the traditional manner from a secure client. The same password can also be used from a non-secure client in a manner highly resistant to eavesdropping attacks. Although more complex than traditional password entry, in our tests 11 out of 13 users had overall success rates of 80% and above, and 12 of the 13 users had 100% success rates after the initial learning stage. The average authentication time is 1-2 minutes, depending on the password length. Like all similar authentication schemes reported to date, this scheme is too cumbersome for general use, but could be useful for special situations and with motivated users.*

Keywords: Authentication, eavesdropping, shoulder surfing, peeping attacks.

1 Introduction

Mobile users often need to authenticate from non-trusted clients to trusted remote servers. In such situations, passwords are extremely vulnerable, because a shoulder surfer or key logger on the client can capture the password and give the attacker full access to the user's remote account. Other authentication mechanisms such as trusted hardware or biometric sensors are not widely available. One solution is a password scheme that allows authentication without disclosing the password to a key logging attack.

We propose a scheme that allows the user to enter a traditional password in a trusted environment and to enter the same password in a secure manner from a non-trusted client. The ability to use the same password in secure and non-secure environments is critical given the user's limited ability to recall passwords [1].

While some work has been done on passwords that are resistant to shoulder surfing [2,3] or to general eavesdropping attacks [4], only one approach allows the reuse of the same password in both secure and non-secure environments [5,6]. However, this approach requires gaze-tracking hardware and is only resistant to shoulder surfing attacks. Our scheme allows the reuse of the same password in secure and non-secure environments with general-purpose hardware.

A common characteristic of all schemes proposed to counter eavesdropping attacks [2-4] is that they are much more complex and time consuming than the traditional password interface. For this reason, we do not view our scheme as a replacement for traditional password authentication. Only in non-secure environments where there are no other options and only with motivated, savvy users is the scheme expected to be practical. Users must understand and accept the tradeoff in complexity and authentication speed in exchange for increased security. It is unlikely that our proposed scheme will ever be used on a large scale by the general public, for example in e-commerce applications. This scheme is more likely to be used by James Bond than by Jim Doe.

2 Related Work

Password-only authentication schemes able to withstand eavesdropping attacks were first mentioned in 1969 by Hoffman [7], but only detailed two decades later in a paper by Matsumoto and Imai [8]. This work and all subsequent approaches reported in the literature rely on requiring the users to carry out a simple hashing function in their head. The hashing function combines a shared secret (known to both the user and the authenticating server) with random information displayed on the computer screen (known to the user, the server and to any attacker observing the authentication session). The nature of the hashing function makes it difficult for the observer to extract the shared secret from the known user response and the known random component. Also, because the correct answer incorporates both the shared key and the random component, the user input from one session cannot be used to authenticate in another session that uses a different random component.

Matsumoto's initial work [8] set the theoretical basis for several protocols that are resistant to eavesdropping that are based on mathematical operations or on tracing a path on a map. The most effective attack strategy involves collecting user authentication information across several sessions, using this data to narrow down the space of possible passwords, and then attempting a brute force attack on the remaining password space.

More recently, Sobrado et al. [9] describe a scheme based on spatial relationships that requires the user to visualize geometrical shapes. The password is a set of graphical symbols that are displayed in random order on the computer monitor. To authenticate, the user must click inside the convex hull determined by the chosen symbols. To achieve sufficient confidence in the identity of the user, several correct click sessions are required for any authentication session. The scheme underwent many generations of improvements to eliminate inequalities in the probability distribution of the correct click location (which has implications on the security of the scheme). In its latest form, the scheme was reported in a recent paper by Wiedenbeck et al. [3]. Study results indicate success rates above 90% for authentication sessions for all users, and 98% for individual click sessions in the authentication process. Reportedly, many of the authentication sessions failed because of a single failed click session. The time required for the authentication was significantly longer than for the conventional password: an average of 10 seconds per click session, which translates in times of up to a minute for a password requiring 6 clicks.

One major drawback of the scheme is that the chosen symbols are rather difficult to distinguish from each other, and that they are not similar to a typical user password. Zhao and Li [10] extended this system so that it may be used with alphanumeric characters chosen from a traditional password instead of graphical symbols.

Another authentication procedure resistant to shoulder surfing is based on mathematical operations (e.g. requiring the user to perform modular arithmetic), and published in a paper by Hopper and Blum [11]. Although this work is widely cited, it does not include a usability component.

Finally, a proposed scheme based on combinations of graphical objects and alphanumeric characters [4] include only a brief mention of the usability study. Users were able to start using the scheme within 15 minutes of the initial exposure.

A common drawback of all the proposed schemes is that they require additional cognitive effort on the part of the user, and that the authentication time is longer than traditional password schemes. The added cognitive load is required to perform the hashing operation in the user's head, and the longer time results directly from the increased cognitive load.

Reusing the same text-based passwords in a traditional password entry scheme and in a scheme resistant to eavesdropping attacks increases usability, because it reduces the number of different passwords a user must memorize. We present the user interface and the usability of such a scheme in the remainder of this paper.

3 Proposed Scheme

We propose a password entry scheme that will allow the reuse of the same password in conventional authentication (when using a secure computer) and in a manner that is highly resistant to eavesdropping attacks (when using a less secure client computer). The scheme we propose has several direct advantages. First, users do not need to remember additional passwords to authenticate from non-secure clients. Second, the password format required by our proposed scheme forces the user into choosing a "strong" and relatively long password, yet makes the password relatively easy to memorize.

When using our scheme from a non-secure client, we assume that the user input to the computer can be monitored, recorded and later processed by the attacker using unlimited computing power to extract the user's authentication information. This situation is very different from the usual applications of encryption, where both the user and attacker have access to computing power. This same advantage of the attacker makes it very difficult to design a user interface that is simple, easy to use and that at the same time has any ability to protect against the attacker.

3.1 Shared secrets to resist eavesdropping

The approach we take to protect the authentication information is to use a simple problem that the user can solve using only mental power. The problem is based on a secret shared between the user and the authentication computer. For an authentication session, the user input is a function of the problem displayed on the screen and the shared secret, which the attacker does not know. Key to this scheme's operation is that, given a problem on the screen, multiple values of the shared secret may lead to the same user input.

As a simple example of this approach, consider a shared secret that consists of the letters "ABC". The screen displays the entire set of possible alphanumeric symbols with a random choice of background (either white or dark) for each symbol. The user input requested might be the number of symbols that are displayed using a dark background, which in this example will be a number between 0-3 depending on the random background colors. Given only this number as the user input, the attacker may find many possible sets of symbols that meet the requirement, and would have no way of knowing which of the sets is the shared secret.

The authenticating server can easily verify whether the user input is compatible with the shared secret. There is no way to know whether the user actually knew the shared secret, or whether some randomly generated input happened to match the expected input. The confidence in the user's identity can be increased by repeating the process. If the user is able to solve several successive problems consistent with the shared secret, the probability of a random guess can be made arbitrarily small at the expense of a longer authentication session.

The attacker has access to the problem displayed on the screen and to the user response. Based on this information, the attacker can reverse engineer the user's input to determine a set of possible values for the shared secret. The security of our scheme rests on having sufficiently many elements of this set. One way for the attacker to narrow down the set of possible shared secrets is to gather user input over several authentication sessions.

3.2 User interface considerations

An important consideration in the usability of the interface is the cognitive load involved in determining the user input, given the shared secret and the displayed problem on the screen. A very simple input (for example, binary valued) is easier to calculate, but gives only 50% assurance that the user actually knows the secret. This requires a large number of repeated trials to ensure high confidence in the authentication. The likelihood that the user will make a mistake increases with the number of trials, overall reducing the chance that the user will authenticate correctly. Additionally, a large number of trials implies a long authentication time.

A similar consideration applies to the complexity of the shared secret involved. A more complex shared secret is more difficult to guess, but the user will have a more difficult time recalling it and operating on it to determine the required input in response to a problem on the screen. A simpler secret allows the attacker to guess more easily.

In a balance of these conflicting requirements, our proposed scheme uses a longer shared secret. We force users to choose shared secrets composed of several triplets of alphanumeric symbols. For each triplet in the shared secret the user must determine the required input, which is another alphanumeric symbol.

3.3 Design rules for creating a shared secret

Our scheme requires that the shared secret consist of an integer number of triplet symbols. In a secure environment, this shared secret is entered as the user password. From a non-secure client, the user will enter one symbol for each triplet using the technique described in section 3.5.

One way to create a shared secret consisting of easy to recall triplets is to choose a series of 3-5 words each containing at least three letters. The shared secret could be a series of the first three letters in each word. The words can form a phrase or may be in a series that is easy to recall for the user. For example, if starting with the phrase "maintain your password secure," the resulting password would be "maiyoupassec" to authenticate from a secure machine. The triplets to authenticate from a non-secure machine would be "mai", "you", "pas", and "sec". Such passwords are generally accepted as difficult to guess [12], unless the user chooses a very common phrase. This approach ensures that each triplet has meaning independent of the rest of the shared secret. Users can hold in their mind one triplet at the time and determine the corresponding entered symbol.

Each triplet corresponds to only one correct entered symbol, but each entered symbol could have been determined from a number of different triplets. In our proposed scheme, if the interface has N possible symbols then the number of possible triplets for a given "entered symbol" is N2 (see section 3.6). For a usable interface, this must be a small number. The protection afforded by this small number is that an attacker cannot tell which of the different possible triplets is part of the shared secret. Most accounts would lock out if the attacker attempts a brute force attack of all possible triplets.

3.4 User interface

In our experiment, only lowercase letters and digits are displayed in random order on a grid of 6x6 symbols. For added security, the scheme could use a larger set of characters, for example both lowercase and uppercase letters, digits and other characters. However, a larger set of symbols would make it more difficult for the user to locate the "password triplet symbols" on the grid. Another disadvantage is that letters would either be capitalized at the start of the word (i.e. the first letter in the triplet), not at all, or in a manner that would require additional recall effort on the part of the user. Although we chose to use a square 6x6 grid, the grid does not need to be square.

3.5 Determining the "entered symbol"

To authenticate, the user enters one "Entered symbol", which we refer to as E, for each group of three consecutive symbols in the "Password Triplet Symbols", which we refer to as PTS. The symbol E is such that it completes a parallelogram on the displayed grid with the PTS. Given any PTS on the grid, there are three different parallelograms that could be formed. Of these three parallelograms, the user must choose the one parallelogram where E is located diagonally opposed to the first symbol in the PTS. To select E the user clicks on the button displaying the symbol or could enter the symbol via the keyboard. The process is shown in Figure 1.

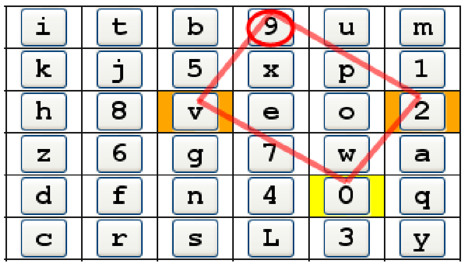

Figure 1. Selecting the Entered Symbol "9", for three Password Triplet Symbols, "0v2".

In Figure 1 the triplet is "0v2" and the fourth symbol to complete the parallelogram is the circled "9". The highlighted parallelogram on the PTS is shown in the figure only to illustrate the scheme, but is not displayed in the actual user interface.

We expect some readers may feel rather confused about the explanation above. Fortunately, there is a sequence of simple rules that can be followed to locate E. Given three alphanumeric symbols, start at the first symbol and count the distance in rows (up or down) and in columns (left or right) to the second symbol. Then, starting at the third symbol, count the same number of rows/columns and in the same direction as before (up or down, and right or left). In Fig. 1, the first symbol is "0". The second symbol, "v", is two columns to the left and two rows above the first symbol. To locate E we start at the third symbol, "2", and count two columns left and two rows up resulting in "9".

In some cases the location of E will fall outside of the displayed grid. When this occurs the user must visualize the displayed grid as a torus that wraps around the edges. An example is given in Figure. 2. The password triplet symbols set is "5oh." If the grid were to extend beyond the alphanumeric squares, then E would be the top position in the second column, two positions above the "h", as indicated. Since there is no button at this location, the user must visualize wrapping the top of the grid back to the bottom (forming a torus) and must click on the "y".

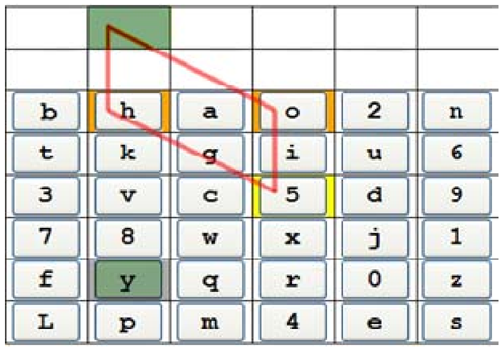

Figure 2. Illustration of the scenario where the entered symbol determined by the password triplet symbols falls outside of the grid of symbols displayed.

3.6 Capability to withstand eavesdropping

We assume that an eavesdropping observer will have full access to the user input to the computer (through a key logger or screen recorder on the user's computer). The attacker may record several authentication sessions and correlate user entered data across all sessions captured. This is the worst case, a case often ignored or dismissed in the literature.

The position of E is dependent on the positions of the PTS. Because these three PTS will be displayed in a different position on the grid every time the user authenticates, E will likely be different for each authentication session. Thus, an attacker must record a large number of authentication sessions to see a repeated arrangement of the PTS. The total number of configurations of the 6x6 grid is of the order of 36! (reduced by the possible symmetries), making it unlikely that the attacker can expect to see an identical configuration in a reasonable time frame.

For the proposed scheme each E is a function of the same three PTS. This "decouples" the password, and allows the attacker to solve for each PTS set separately. Fortunately, for any given E selected by the user there is a large number of PTS that could have determined the selection of that symbol. The degrees of freedom are the positions of two of the symbols in the triplet (the third one is uniquely determined by the position of the first two symbols and the one selected by the user). With two degrees of freedom in an array of 36 symbols, the number of possibilities is 36*36 = 1296.

Armed with only the knowledge of where the user clicked in one particular authentication session, the attacker will not be able to authenticate in place of the user in a subsequent session. Indeed, the next time the attacker attempts to log in, the distribution of symbols on the log in screen will be different than in the previous session. The new arrangement of the PTS in the grid will correspond to another E and the attacker will need to know the actual triplet, not the previously selected symbol to authenticate at this time.

However, the strength of the scheme is limited under repeat observations. An attacker who records multiple authentication sessions, each with a different, random distribution of symbols on the screen will know that the actual triplet of password symbols is in the intersection of possible "triplet password symbols" leading to the E observed in the authentication sessions. For a given E in one session, the number of possible triplets that could have led to that symbol is 1296. If the attacker observes the same E across two sessions, the number of possible triplets that could have led to the two "entered symbols" is reduced to 36. For three or more sessions, the attacker can on average determine the exact triplet in the password.

The situation in reality is not as dire as this summary analysis might indicate. First, each password will comprise a series of 5-8 E values that correspond to the same number of PTS sets. To uncover the user password, the attacker will need to observe E in several sessions, and will need to run the intersections above for each set of PTS. The level of sophistication required to mount such an attack is significantly higher than that required by current phishing attacks. This complexity will lead to a decrease in the number of attacks, but will not deter all attackers.

There are several solutions to handle the determined attackers that might target the limitations of the proposed scheme. Because the vulnerability arises when the same attacker is able to capture several successful sessions from the same user, one protection mechanism is for users to not use the same non-secure computer to log in more than once. As a second solution, the scheme can be made more secure (although even less user friendly) by using 4-tuples or 5-tuples to determine the click symbol. The cognitive load in such cases would be even greater, which limits the usefulness of such an approach. Finally, a third approach allows the user to make errors in the authentication session. Such a "noise" component will i) make it significantly more difficult for the attacker to uncover the PTS corresponding to the observed E symbols, ii) make it easier for the user to authenticate (if the system tolerates some of the user errors) and iii) only slightly increase the authentication time for a given confidence level.

In order to achieve the benefits described above in terms of resistance to attack, the proposed password entry scheme requires a significantly increased cognitive load on the users when compared to the conventional password entry scheme. In addition to the challenges to recall the password, users of our proposed scheme will need to be able to segment the password in groups of three symbols and recall which group they are currently dealing with.

To assess the challenges imposed by our scheme we conducted a small usability study that examined the ability of users to locate the correct "entered symbol," given a triplet in the password.

4 Experimental Methodology

A total of 20 users were invited to participate in the main study. All users were contacted by email and sent a personal access code that allowed us to identify their data. Users were asked to test the software as a favor, and not to spend more than 30 minutes on testing, regardless of the outcome of the testing. No incentives were offered for completing a certain number of sessions, nor for achieving higher accuracy. Users were advised to practice the scheme only until they felt comfortable with it.

In the email contact users were directed to follow a link to a web site where the authentication scheme could be tested online. The web site includes a validation screen (where participants were instructed to enter their access code) and a usage scenario of logging into a secure server from an unfamiliar Internet café as motivation for the scheme.

Out of the 20 users invited to participate in the study, 13 provided data. According to personal follow-up questions, the other users did not get a chance to even consider the application. Almost all 13 users are college students or have completed college, and four have doctoral degrees. They all use computers at work and have a wide range of levels of familiarity with computer technologies, but none of the users had been exposed to our password application before the study. Many of the users were not "techies," but rather administrative staff, techno-phobes, or health care workers. Five of the users are male and eight are female. The mean age is 41 years, and the age range is 24-62.

The users were given the following instructions and a graphical example similar to Fig. 1: "Given three alphanumeric symbols, count from the first symbol to the second symbol -- the distance in rows (up or down) and in columns (left or right). Then, starting at the third symbol, count the same number of rows/columns and in the same direction as before (up or down, and right or left). "

For each user attempt, the computer generated a random "password triplet symbols" set and displayed this set on the screen, along with a random grid of 6 x 6 buttons with lowercase letters and digits. The user was instructed to locate the appropriate button with the "entered symbol" and to click on it. We collected the timestamp of the moment when the "entered symbol" was selected, as well as information about the correct symbol and the actual symbol selected. Because most of the attempts occurred in sequence, we used the difference between timestamps as a measure of the time users spent authenticating.

5 Results

Most users found the scheme difficult to understand at first, but they quickly learned to locate the "entered symbol" for a given triplet. In the discussion below, each triplet entry point counts as one authentication session (the analog of the click session in Wiedenbeck [3]). In practice, a secure password would require 3-5 triplet points, corresponding to the 3-5 words that make up the password. Having to handle several triplets in sequence will likely further reduce the figures of success rate we report and further increase the login time.

Summary data for all users is included in Table 1. Among the 13 users, only one had an overall success rate below 50% (49% for User ID 6 in the table).

The time users took to complete each authentication attempt is significantly longer than the typical 2-10 seconds it takes to type in a traditional password. Users took an average of 20 seconds per triplet, similar to time reported by Wiedenbeck [3]. The range of authentication times is 10-30 seconds per triplet. In a practical situation, for a password requiring 3-5 clicks, a user would need to spend between 30 seconds and 2 minutes to validate their password. This is an acceptable value for users according to Wiedenbeck [3] (ten seconds per click session) and Tan et al. [2] (50 seconds for an average password). Even if the time is too long for daily use, it might be acceptable for authenticating from a non-secure client.

It is reassuring that some of the users were able to authenticate in a much shorter time than the average. In

User ID	1	2	3	4	5	6	7	8	9	10	11	12	13
Nbr. attempts	53	34	25	19	10	76	195	66	81	60	18	21	11
Nbr. successes	51	28	25	16	8	37	171	60	64	60	16	20	11
Success rate	96%	82%	100%	84%	80%	49%	88%	91%	79%	100%	89%	95%	100%
Mean time per click [s]	18.6	18.9	21.8	33.7	20.5	20.3	13.9	16.8	19.5	10.3	27.9	24.1	18.1
Stdev./click [s]	9.5	12.1	10.8	17.1	7.0	14.6	10.1	6.5	10.8	2.8	15.6	10.7	11.5
Total time spent [min]	15.2	9.5	8.0	7.9	2.7	22.4	44.1	17.1	25.3	10.1	7.5	6.8	3.0
Success rate last 10 triplets	100%	100%	100%	90%	80%	100 %	100%	100%	100%	100%	100%	100%	100%
First series of 10 successes	3	20	1	2	n.a.	58	24	18	52	1	3	10	1

Table 1. Summary of results for all 13 users

some cases users took less than one second per triplet and almost all users were able to correctly choose at least some of the click points correctly in 10 seconds or less. User #10 required an average of 10.3 seconds per triplet.

There was some statistical correlation between average time and success of the authentication. We ran a paired t-test comparing the mean times for all sessions with the mean times for only the successful sessions for each user. The difference was 0.5 seconds (p=0.098). Interestingly, the average time for correct attempts was shorter than the average time for all attempts.

User ID #7 provided close to 200 click points, allowing us to see a trend in decreasing authentication time. As shown in Figure 3, there is a trend for the authentication time to approach 10 seconds, although this user initially started with much slower authentication times. The overall average time for the user is 13.9 seconds.

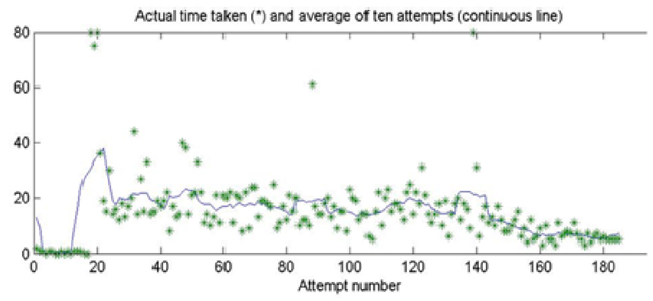

Figure 3. Time per click in seconds for a “high persistence” user. Actual times displayed are clipped at 80 seconds.

We attribute the low initial authentication times (bottom left corner of Figure 3) to attempts to understand how the data entry worked. It is unlikely that the user actually considered the possible click point, given the extremely short authentication time taken. The success rate is close to zero for these initial points.

We expected a rather steep learning curve, so we focused on the success rate after sufficient learning has occurred, not on how quickly users would be able to master the scheme. Our data indicates that after the initial learning period most users do achieve almost error-free performance. To quantify the situation we measured success for the last 10 attempts; all but two of the 13 users achieved 100% success rates. The other two users had 90% and 80% accuracy, respectively for these last 10 attempts. Choosing a different metric, for example the success in the last 5 or 15 attempts leads to similar results. Notably, User ID #6 with an overall success rate of 49% has a 100% success rate over the last 10 attempts.

What would matter in a realistic authentication scenario is whether users would be able to sustain good accuracy in selecting the entered symbols, E, over several sessions. To measure this, we averaged the user success rate over a sliding window of 10 consecutive attempts. The last row in Table 1 shows the number of attempts each user took to achieve a windowed success rate of 100%. This 100% windowed success rate is reached when the user is able to achieve a series of 10 consecutive successful E’s in a row. Clearly, each user must have at least 10 attempts for such a metric to be defined. The metric is relevant, because a user who can select 10 consecutive E’s correctly is able to authenticate with reasonable accuracy if the password includes 3-5 entered symbols. For clarity, the entries in the last row in Table 1 are measured after the 10th attempt. For example, User ID #3 achieved 10 correct sessions as soon as completing the 10th attempt (all ten first attempts were correct). Seven of the users achieve 10 consecutive correct entries within three attempts of their 10th. Another group of four users reach 100% windowed success within 10-25 attempts past their first 10, and a group of two “laggards” take more than 50 attempts to reach 100% success.

For all users, the success rate windowed over 10 consecutive attempts is a function that reaches 100% relatively fast, within at most 50-60 entries, and for some users even from the first few tries. This windowed success rate does not go up monotonically, but once it reaches 100% it remains above 70% for all users. This indicates that users have different learning speeds, but that once they "get" the scheme, the error rate is relatively low.

6 Discussion

Based on observations to date, the proposed password entry scheme has the potential to be usable among well-trained, savvy, and well-motivated users. All users found the scheme difficult at first, but once users learned how to apply the counting technique, "it all clicked in" and the process appeared much easier.

The ability to recall a password is a critical factor for authentication schemes, and many users have difficulty with strong passwords [13]. Our scheme compounds these difficulties with the need to recall the actual authentication scheme. We do not have any data yet on how well users' ability to use the scheme can persist over time. Given the reported experience with understanding of the scheme "clicking in," we expect the scheme to be memorable.

7 Conclusions

Users who have to authenticate from non-secure clients must understand that they need to trade off convenience for security. We propose and evaluate a scheme that allows users to authenticate with the same password in a non-secure environment, without disclosing the password to an eavesdropping attacker. To afford such protection, the scheme is necessarily more complex, and requires a longer authentication time. Informal interviews with test users indicate that they find the authentication scheme to be tedious, but usable.

Users found the scheme complicated at first, but reported that it became easy to use after practice. For the scheme to be practical, users must devote practice time in a secure environment ahead of time. Our testing indicates that most users can get comfortable and proficient with the scheme within 30 minutes.

Both the high cognitive load and the lengthy authentication time make the scheme unlikely to be widely adopted for the general population, for example, authentication into e-commerce applications. On the other hand, we expect the scheme to be acceptable for specialized applications and motivated users.

8 References

[1]. Tari, F., Ozok, A. and Holden, S. *A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords.* Pittsburgh, PA, July 12-14 : ACM Press, 2006. Proceedings of the Second Symposium on Usable Privacy and Security. pp. 56-66.

[2]. Tan, D., Keyani, P. and Dzerwinski, M. *Spy resistant keyboard: more secure password entry on public touch screen displays.* Canberra, Australia : s.n., 2005. Proceedings of the 19th Conference of the Computer-Human Interaction Special Interest Group of Australia. Vol. 122.

[3]. Wiedenbeck, S., et al. *Design and evaluation of a shoulder-surfing resistant graphical password scheme.* New York, NY : ACM Press, 2006. Proceedings of the Working Conference on Advanced Visual Interfaces. pp. 177-184.

[4]. Man, S., et al. *A password scheme strongly resistant to spyware.* Las Vegas, NV : s.n., 2004. International Conference on Security and Management. pp. 94-100.

[5]. Hoanca, B. and Mock, K. *Secure graphical password system for high-traffic public areas.* San Diego, CA : s.n., 2006. Eye Tracking Research & Applications Symposium.

[6]. Kumar, M., et al. *Reducing shoulder-surfing by using gaze-based password entry.* Pittsburg, PA : ACM Press, 2007. Proceedings of the 3rd Symposium on Usable Privacy and Security, SOUPS '07. Vol. 229, pp. 13-19.

[7]. Hoffman, L.J. *Computers and privacy: A survey.* 2, s.l. : ACM Press, 1969, ACM Computing Surveys, Vol. 1, pp. 85-103.

[8]. Matsumoto, T and Imai, H. *Human identification through insecure channel.* 1991. EUROCRYPT. pp. 409-421.

[9]. Sobrado, L. and Birget, J.C. Graphical Passwords. *The Rutgers Scholar.* [Online] vol. 4, 2002. [Cited: September 12, 2006.] http://rutgersscholar.rutgers.edu/volume04.

[10]. Zhao, H and Li, X. *S3PAS: A Scalable Shoulder-Surfing Resistant Textual-Graphical Password Authentication Scheme.* 2007. Proceedings of the 21st IEEE International Conference on Advanced Information Networking and Applications Workshops.

[11]. Hopper, N. and Blum, M. A Secure Human-Computer Authentication Scheme. [Online] CMU Tech Report CMU-CS-00-139. [Cited: February 14, 2005.] http://www.andrew.cmu.edu/user/abender/humanaut/links.html.

[12]. Kuo, C., Romanosky, S. and Cranor, L. *Human selection of mnemonic phrase-based passwords.* Pittsburgh, PA : ACM Press, 2006. Proceedings of the Second Symposium on Usable Privacy and Security (SOUPS '06). pp. 78-78.

[13]. Sasse, M., Brostoff, S. and Weirich, D. *Transforming the "weakest link": a human-computer interaction approach to usable and effective security.* 3, July 2001, BT Technology Journal, Vol. 19, pp. 122-131.