Modeling Password Entry on a Mobile Device

Melissa A. Gallagher (melissa.gallagher@rice.edu)
Department of Psychology, MS-25
Houston, TX 77005 USA

Michael D. Byrne (byrne@rice.edu)
Departments of Psychology and Computer Science, MS-25
Houston, TX 77005 USA

Abstract

Password authentication is a widely deployed security feature on desktop and mobile systems. Inputting complex passwords on mobile devices can be an onerous task. The composition of the passwords creates a unique challenge for people to input as not all characters are displayed on the keyboard at the same time, forcing the user to switch between multiple screens. The results from a previous study informed an ACT-R model of password input on mobile devices. The timing data generated from the model fits the experimental results well. The strategy that the model employs compliments the results from the experiment providing further information into the strategy subjects employed. Validated models of password input on mobile devices are an important tool that can aid designers in usability testing and security professionals when creating new password policies.

Keywords: Passwords; mobile typing; touchscreens; human factors; useable security.

Introduction

Twelve characters long, one number, one uppercase letter, and one special character; password must contain at least two of the following types of characters: letters, numbers, and symbols; these are just two examples of enforced password policies meant to make passwords more secure. Many systems now set a minimum length as well as force users to include non-alphabetic characters in their password. The policies are enforced by the system because they give the passwords high entropy. Passwords are considered high-entropy if they are long and contain a variety of character types. This combination makes them harder to crack using a brute force attack. As the computational power of computers increases, systems are increasingly enforcing password policies that make them high-entropy. These policies vary from system to system and make passwords not only difficult to remember but difficult to input. At the same time users are creating more and more accounts with different systems and must remember an ever-increasing number of passwords (Florencio & Herley, 2007). But there is recognition that the user is an integral part of the security of the overall system (Adams & Sasse, 1999). Being able to test the effects of new password policies prior to enforcing them would be beneficial to security professionals as well as users of the system.

Inputting passwords on mobile devices presents a unique challenge for the users not present on desktop systems. Commonly typed characters are always visible on physical keyboards; on mobile devices characters are on multiple screens that the user must shift between. Navigating between different screens not only increases the number of taps before the character can be input but it also requires the user to remember the keyboard screen, or screen depth, in addition to the location of the character. Now users must recall the password, keep track of a character's position within a password, its spatial location on the keyboard, and its relative screen depth. This becomes even more complicated if the current character is available on multiple screens. Passwords of longer length provide more opportunities for input error. These differences can place significant perceptual-motor and cognitive demands on users. Figure 1 shows the layout of the keyboard at the different screen depths on an iPhone.

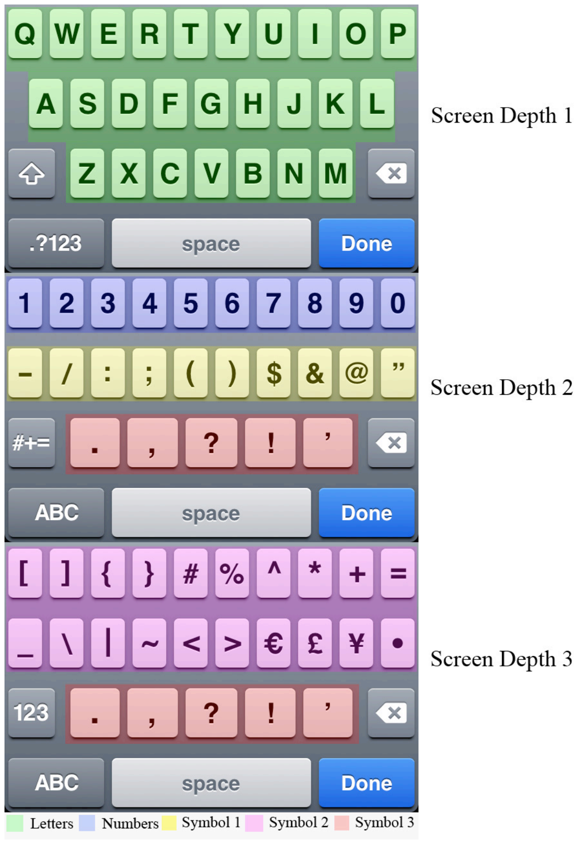

Figure 1. The categories of keys based on their screen depth and type on an iPhone

ACT-R (Anderson, 2007) is a multi-domain cognitive architecture for simulating and understanding human cognition and performance. ACT-R's current typing ability is comparable to that of a moderately skilled touch typist. Moderately skilled is defined as typing about 30-40 WPM and knowing the location of the keys without having to look for them but not performing as rapidly as an expert typist (John, 1996). Das and Stuerzlinger (2007) built an ACT-R model simulating expert text input on a cellular telephone with a 12-button telephone keypad using multi-tap as the input method. Multi-tap is an older system of text entry where the user presses a key to cycle through the letters associated with that key; for example pressing "6" twice would produce an "N." With the proliferation of smartphones, this older input method is much less common today as smartphones supply full keyboards as well as other methods for text input. A validated ACT-R model of text input on a smartphone is not available. In making progress toward building a working model of password input on mobile devices, a model of the typing portion of password input was built in isolation from the memory component. Since the models are expandable, as more aspects of the task need to be modeled, the model can change to incorporate errors and the memory component of the task.

Modeling

An ACT-R model was built to model the input speed from the password transcription typing experiments in Gallagher (2014). The task presented subjects with a string similar to a high entropy password. The string was always present at the top of the screen. Subjects were instructed to type the string exactly as it appeared as quickly and accurately as possible. Two of the experimental results heavily influenced the strategy employed by the model. The first result was that as subjects progressed through this task their input speed increased because they spent less time on the page before inputting the target character. The second result was that throughout the experiment subjects did not navigate efficiently between keyboard screen depths. The model was built to provide insight into some of the performance aspects of the task that were not explained by the subject data alone. One question was why did the time spent on the page before symbols were typed remain slower than non-symbols. A second question was how were subjects searching through keyboard screens to find the characters they were less familiar with.

Method

Data Modeled

The model was constructed based on the subjects who interacted with the smartphone using one finger to input text. We took interkey interval (IKI) as our primary dependent measure rather than the more coarse measure of words per minute. This has the advantage of increasing the constraint on the model, since matching only global performance obscures details of how that performance arises. Furthermore, this allowed us to focus on error-free performance, since WPM also includes error correction, which was beyond the scope of this initial inquiry.

There were six character categories: Lowercase, Uppercase, Number, Symbol1, Symbol2, and Symbol3. These categories were determined based on which screen depth characters were on, whether a shift key was required, and their type. The letters were in the categories Lowercase and Uppercase, and were distinguished by case due to the shift key having to be pressed prior to inputting an uppercase letter. The category Number represented the numbers, which are on screen depth 2; Symbol1 represented the symbols that were visible only on the same page as the numbers. The decision to separate these two groups was because we hypothesized that subjects would be more familiar with where the numbers were on the screen and in relation to each other than they would be with the symbols. Symbol2 represented the symbols that were visible only on screen depth three. Symbol3 were the symbols that were visible on both screen depth 2 and 3. Figure 1 shows the location of the key categories.

Materials

Since there is no way to interface ACT-R directly with the iOS simulator, a custom environment was built in Common Lisp for the model to interact with. The model environment mimicked the mobile application used by the subjects. The arrangement, space, and size of the interface elements were the same. Figure 2 shows the two interfaces. The interface elements that are unique to the mobile application, like the touch keyboard and the masking password field, were reconstructed as accurately as possible in the model environment.

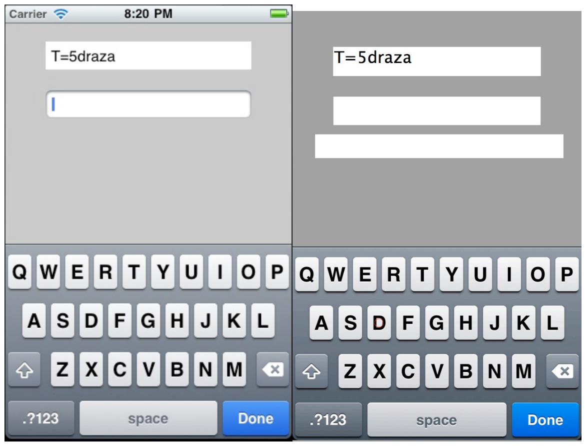

Figure 2. The layout of the iOS application (left) and the simulated Lisp environment (right)

The keyboard was built so that the visible key size was the same as the visible key size in the mobile application, but the functional key size extended beyond that. Since the exact functional key size is not publicly available information, the simulated keyboard evenly split the difference between adjacent keys. Another feature unique to the mobile application that was reconstructed was the visual change on a key press. When a key is pressed an enlarged version of that key is displayed above the regular position of the key; upon release of the press the enlarged version of the key disappears. In the mobile application, the password field shows the character for a short amount of time after it is typed before masking it with an asterisk. If characters are typed into the password field in rapid succession then the characters are masked as the next key is input even if the normal time before masking has not expired. The password field used in the model environment recreated this behavior and used a time of one second before masking the most recently typed character. The mouse device module in ACT-R was used as a stand-in for a finger in touch interactions.

Design

The model was given a moderate amount of knowledge in declarative memory. For all characters on the keyboard, the model knew if they were letters, numbers, or symbols. The model knew the locations and screen depths of all the letters and numbers. The model only knew the location and screen depth of symbols that were presented to the subjects in the practice blocks of the experiment. For all other symbols, the model did not start with the knowledge of their location or screen depth. The final part of the model's knowledge was which keyboard change key needed to be pressed to navigate between different screens. These assumptions were based on most subjects' familiarity with a QWERTY keyboard, the relationship between numbers, and the experienced gained from doing the practice block.

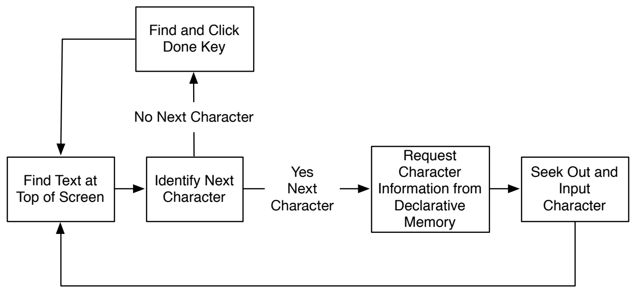
Find and Click Done Key
No Next Character
Find Text at Top of Screen
Identify Next Character
Yes Next Character
Request Character Information from Declarative Memory
Seek Out and Input Character

Figure 3. The overall model strategy

The overall strategy that the model employed to input the stimuli was straightforward, a flowchart of it is shown in Figure 3. At the beginning of each trial the model would determine where the stimulus was by picking out the text that was at the top of the screen. The model would then shift visual attention to the text, identify what the first character was, and store it in the goal buffer. The model would then make a recall request to the declarative memory for all available information regarding that character. Either just the type of the character would be recalled or the type of the character, the screen depth, and the location of the key. The model would then seek out and input that character. Once the character was input the model would shift visual attention back to the stimulus and pick out the next character. This process repeated for all the characters in the stimulus. After the last character was input, the model would shift attention back to the stimulus and identify there were no more characters and seek out the done key. To identify the done key the model would both shift visual attention and the mouse in parallel to the keyboard key in the bottom right of the screen, identify it was the done key, and click on it.

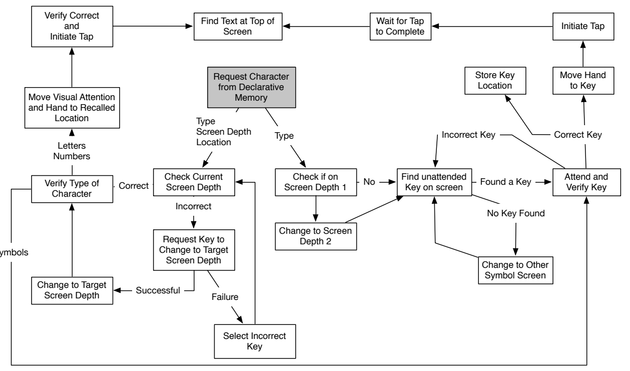
Verify Correct and Initiate Tap
Find Text at Top of Screen
Wait for Tap to Complete
Initiate Tap
Request Character from Declarative Memory
Store Key Location
Move Hand to Key
Move Visual Attention and Hand to Recalled Location
Type Screen Depth Location
Type
Incorrect Key
Correct Key
Letters Numbers
Verify Type of Character
Correct
Check Current Screen Depth
Check if on Screen Depth 1
No
Find unattended Key on screen
Found a Key
Attend and Verify Key
Incorrect
Change to Screen Depth 2
No Key Found
Symbols
Request Key to Change to Target Screen Depth
Change to Other Symbol Screen
Change to Target Screen Depth
Successful
Failure
Select Incorrect Key

Figure 4. The process the model used to seek out and input a character. The grey box is where the process starts.

The different strategies for seeking out and inputting the characters were based on their category and whether or not the model knew where the key was located. Figure 4 illustrates the process and corresponds to the box "Seek Out and Input Character" in Figure 3. The location and screen depth of the letters and number keys were always recalled correctly. The model took an aggressive approach to inputting letters and numbers by doing as many actions in parallel as possible. After recalling the character, if the model was on the correct screen depth it would move both visual attention and the mouse to the key location in parallel. Once visual attention shifted to the key the model would verify that it was the correct key and as soon as the hand reached the key a press action was initiated. Once the press action was initiated, the model would shift visual attention back to the stimulus to determine the next character. For keyboard change keys as well as the shift key the model would move the mouse and visual attention in parallel. When changing the keyboard the model had to wait for the press action to complete and the keyboard to change before it could shift visual attention. When the target character was a symbol of a known location the model took a more conservative input approach and the process was similar but conducted in a serial manner. If the model was at the correct screen depth it would shift visual attention to the key first, verify it was the correct key and then move the mouse toward the key and initiate the press action after arriving. The model waited until the press action was complete before shifting attention back to the stimulus to determine the next character. If the model was searching for a symbol when the key location was not known it would start the search from the first page of symbols it encountered and randomly select an unattended key and compare it to the target character. This process continued until the target key was found or there were no more unattended symbols. If there were not more unattended symbols it would switch to the next page of symbols and start searching there. On screen depth two it did not search through the numbers. If the model started on screen depth one it would switch to screen depth two and start the search there. If the model started on screen depth two or three it would start searching on that page and switch to the other one if it was unable to find the key on the starting page. Once the model successfully located the symbol the location and screen depth were added to declarative memory.

To navigate between screens the model had to correctly recall the key necessary to switch between the current screen and target screen. If the model was unable to recall the correct key it would select the incorrect key. The majority of the extra taps between categories subjects made occurred when the previous character was a number or a symbol and the target character was also a number or a symbol. The number of transitional taps to and from letters was closer to the minimum. Due to the difference in number of taps, the chunks represented by those transitions involving letters started with higher base-level activation than the chunks representing the transitions between number and symbols.

There are three processes in the model that vary in the amount of time they take to complete. The first two are related to the recall of chunks from declarative memory. Recall times for this model are based on base-level activation and a random noise component. The chunks representing character keys started with a higher level of activation and recall for them never failed. The amount of time to recall did vary based on the activation decay and the random noise component. For the chunks representing the character transitions the initial activation started lower so that they would not always be recalled. The time to navigate between screens was influenced by recall time as well as success or failure of recall. The third source of variation is the visual search time when a symbol's screen and location are not known. The preceding character determines the page the model is on before it starts starting searching for a symbol and the order it examines keys is random, the amount of time it takes to find a symbol varies for each run.

To approximate touch screen interaction we used the mouse in place of the finger, similar to Salvucci, Taatgen, and Kushleyeva (2006).

Results

To be able to compare the subject data and the model data, the procedure from Byrne (2013) was used determine the number of times the model needed to be run to build a 95% confidence interval within 5% of the mean interkey intervals. The model was run 20 times to estimate the coefficient of variation for each of the interkey interval. The largest coefficient of variation was used in the computation, and the minimum number of model runs was determined to be 93. One hundred model runs were performed.

The interkey interval for each pair of categories for the model and the subjects can be seen in Figure 5 and Figure 6. The matched pairs of the model and subject data are shown in Figure 7. The results of a simple linear regression indicated that the model was able to predict 88% of the variance (R^2 = .88, subject = 0.69 * model + 0.58). The mean absolute deviation of the model from the subject data was 260 ms or 15.7%. Figure 8 and Figure 9 show the average number of taps the model and the subjects made transitioning between character categories. A simple linear regression indicated that the model was able to predict 90% of the variance (R^2 = .90, subject = 0.95 * model - 0.15). The mean absolute deviation of the model from the subject data was 0.22 taps.

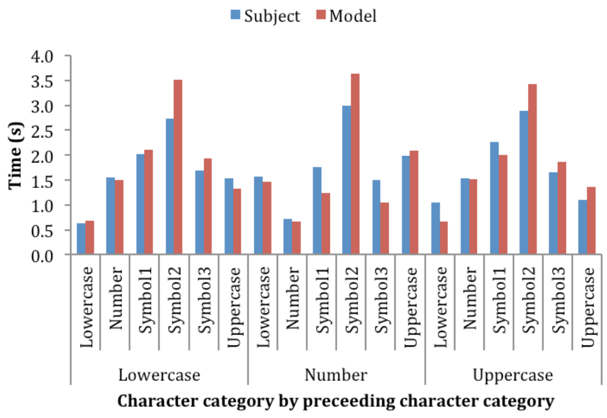

Figure 5. IKI for transitions from Lowercase, Number, and Uppercase

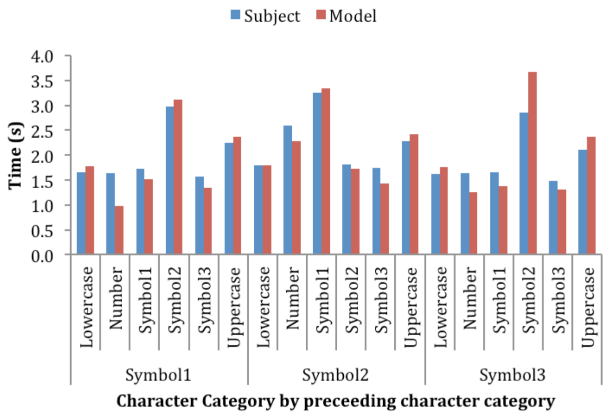

Figure 6. IKI for transitions from Symbol1, Symbol2, and Symbol3

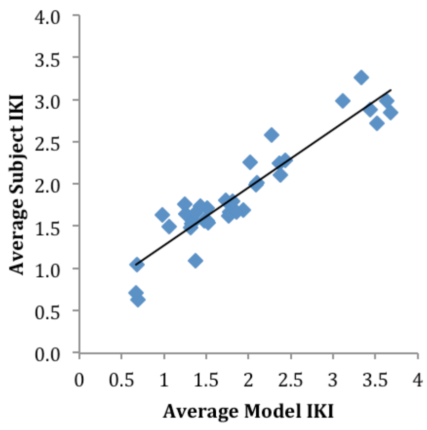

Figure 7. Matched Pair IKI between the subjects and the model

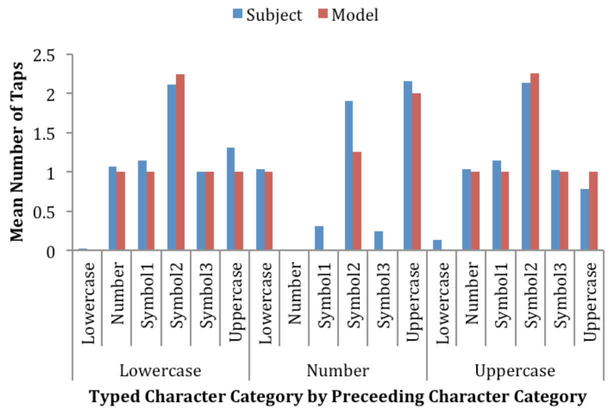

Figure 8. Average number of taps between character categories

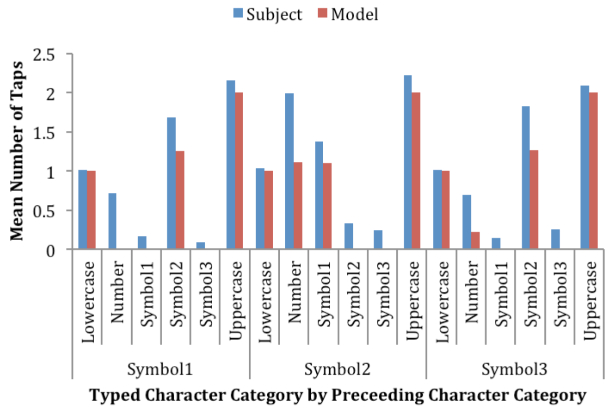

Figure 9. Average number of taps between character categories

Discussion

The strategy that the model uses highlights the three main reasons why inputting high-entropy passwords on mobile devices is slow. The first is different strategies are employed for different character categories. The model and subjects are able to efficiently input characters they are familiar with but take a more conservative approach when inputting symbols. The second reason is that the model does not have knowledge of where the symbols are and has to locate the symbols the first time they are input. The third reason is not being able to navigate efficiently across screen depths. Like the subjects, the model does not navigate efficiently across pages and will not always take the shortest path to the correct page.

These deficiencies in task performance highlight ways that passwords could be structured to improve task performance. Concerning which symbols are selected, using ones on the first page reduces the amount of time spent searching through keyboard screens. Rather than using symbols that subjects are unfamiliar with, symbols that are input frequently could be used so that subjects are more likely to remember their location on the screen. While using high-frequency symbols subjects may eventually employ a less conservative approach when inputting the symbols as they grow more accustomed to it. With regard to inefficient navigation, not using characters from the different symbol screens in sequence would be beneficial because it would eliminate the need to navigate across screens. If the user could modify the keyboard layout, they could place the symbols they would like to use more frequently on the first screen. This would aid them in search time, as they know which symbols they use most and could give them priority. These recommendations for structuring passwords and keyboard designs could be tested with the model to determine if there are performance benefits.

For example, take the passwords *Af_3+2=5_Fa!* and *aAfF235__+=!*. They are both twelve characters long, comprised of the same characters, and meet the requirements of having at least two characters of each type. The first one was created using recommendations for memorability, *A*ddition *f*act _ *3+2=5* _ *F*act *a*ddition*!*. The second one rearranged the characters so that they were in order of screen depth. To predict input time without the variability introduced by visual search the model was modified so that it knew the locations of all the symbols. With this modification the model predicts an input time of 24.57s in landscape and 24.12s in portrait for the first version and 15.99s in landscape and 15.59s in portrait. For the first password the model inefficiently navigated on two transitions but never inefficiently navigated on the second version. Arranging the characters in an order that allows them to be input in the most efficient manner gives a savings of almost 10 seconds.

Future Modeling Directions

There are a number of steps that could be taken to improve the model's performance in the remaining categories. When typing a Number the model is faster when coming from Symbol1 and Symbol3 than the subjects are. One of the reasons that this happens is because the model never misidentifies the page that it is on while subjects do and will navigate off of screen depth two even when they do not have to. When typing a character in Symbol2 the model is slower than subjects for all preceding character categories except Symbol2. One of the reasons that this could be happening is the model is using completely random visual search. There are six symbols that are part of the category Symbol2 that are pairs, [ ] { } and < >. If a subject knows the location and screen depth of one of members of a pair, they can use that information to more quickly find the other member of that pair instead of blindly searching for it. Additionally if subjects are searching and find one half of the pair they can use this as a cue to find the other half where the model just continues searching randomly.

In advancing the model forward the first major change to make would be to have the model make errors when inputting the stimuli. This can likely be accomplished by turning on motor movement noise in ACT-R. With noise on the model would not always successfully acquire the target key and would also not always land in the center of the key. This would introduce the most common kind of errors, adjacent key errors. The subject data indicates that while not all errors are caught and corrected the majority of them are. Therefore the model's strategy would need to change so that sometimes it would verify the input. After inducing the model to make errors, the next step in development would be to branch out to the other input styles and devices. Additionally, there may be variations in subject strategy. Subjects may keep more than one character in working memory at a time instead of referring to the stimuli after each character is typed.

Additionally this model only used an iPhone. Further research needs to be done to be able to generalize to other iOS devices and platforms, e.g., Android and Windows Phone, and the variety of devices they run on. Using Android phones. Now that the size of iPhones has increased and alternative keyboards are available for iOS, these new features can be tested to see if they provide any advantage when typing passwords.

Conclusions and General Discussion

Although mobile device keyboards were designed to be similar to physical keyboards, they are not the same and many of the limitations of the mobile device make typing passwords slower. One of the main factors in the slow input speed is inputting the symbols. While the model learns the location of the symbols the initial act of searching through the keyboards screens is time consuming. Compounding the slow down is the conservative approach taken during the typing process to ensure accuracy. Since the model does not navigate between the symbol pages and number pages efficiently it could be helpful for passwords to not require symbols and numbers in sequence. Having a working model of password input on mobile devices has a number of benefits. When presented with novel passwords, the model can make predictions of password typing time. This is especially useful because as new password policies are generated they can be tested to see if they are detrimental or beneficial to improving input speed. In addition to different password policies, different keyboard designs can be tested prior to implementation. Typing the password is only one component of authentication and needs to be incorporated with work that looks at the cognitive component as well.

Acknowledgments

This research was supported by the National Institute of Standards and Technology under grant #2012-NIST-MSE-01. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of NIST, the U.S. Government, or any other organization. The authors would like to thank Clayton Stanley for his help with the Lisp programming.

References

Adams, A., & Sasse, M. A. (1999). Users are not the enemy. *Communications of the ACM*, 42(12), 40-46.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Byrne, M. D. (2013). How Many Times Should a Stochastic Model Be Run? An Approach Based on Confidence Intervals. *In The 12th International Conference on Cognitive Modeling*, 445-450.

Das, A., & Stuerzlinger, W. (2007). A cognitive simulation model for novice text entry on cell phone keypads. *Proceedings of the 14th European conference on Cognitive ergonomics: invent! explore!*, 141-147.

Florencio, D, & Herley, C. (2007). A large-scale study of web password habits. *Proceedings of the 16th international conference on World Wide Web*, 657-666.

Gallagher, M. A. (2015) Modeling Password Entry on Mobile Devices: Please Check Your Password and Try Again, Doctoral Dissertation, Rice University, Houston TX.

John, B. E. (1996). TYPIST: A theory of performance in skilled typing. *Human-computer interaction*, 11(4), 321-355.

Salvucci, D. D., Taatgen, N. A., & Kushleyeva, Y. (2006). Learning when to switch tasks in a dynamic multitasking environment. *Paper presented at the Proceedings of the seventh international conference on cognitive modeling,* 268 –273.