

# Non-Visual Passcodes for smartphones

Youngmin Lee, Hyunjoon Lee, Dale Noh

University of Washington

Department of Computer Science & Engineering

{lym737, hjee513, styner}@cs.washington.edu

## Abstract

Disability should not be a reason of preventing using the technology. Disabled people should have equal opportunity to use the technology as much as non-disabled people do. However, many technologies are not accessible to disabled people. One of them is smartphone. In this paper, we will introduce the two prototype non-visual passcode Android applications called *BPass* and *GPass* we developed for blind people. We made these applications usable without vision so blind people can use them.

**Keywords:** Accessibility, blind, touch screens, gesture, gesture recognition, security, smartphone.

**ACM Classification Keywords:** D.4.6. [Operating Systems]: Security and Protection - *access controls*; H.5.2. [Information Interfaces And Presentation]: User Interfaces - *input devices and strategies*; K.4.2. [Computers And Society]: Social Issues - *assistive technologies for persons with disabilities*.

**General Terms:** Human Factors, Security

## Introduction

Despite the number of blind people who use the smartphone has increased, smartphones are not designed blind-friendly. Although many software developers have developed applications that help the usage of smartphone for blind people, there are still a lot of smartphone functions inaccessible to them. One of smartphone function that is not accessible to blind people is the smartphone lock screen. The lock screen is highly necessary for the smartphone since in nowadays, people put a lot of personal information in their phone. If any people other than the phone owner can access to the phone, some serious problems can happen such as privacy intrusion or personal information theft. To prevent these problems, smartphone has lock screen with the passcode. To unlock the phone, the user should input the correct passcode. Unfortunately, the current passcode function in smart phone is almost impossible to use without vision.

Figure 1-1<sup>1</sup> and 1-2<sup>2</sup> are two examples of current passcode system in smartphone. To solve the passcodes, you have to click the buttons or drag between buttons on the screen. Since these buttons on the screen are not physical, without seeing the screen, it is really hard to find the correct buttons. The passcode should be easy and fast to solve to give the user immediate access to the phone but for blind people, it would take long time to solve the passcodes or even impossible to solve. In this reason, the blind smartphone users do not use this current lock screen function in their phones and their phones are left unprotected from others.



Figure 1-1, 1-2. Current passcodes in smartphone

To address this problem, we developed two prototype non-visual passcode applications that are much easier to use for blind people than current passcodes and, at the same time, keep the security so that only the phone user can access to his/her phone. Blind people can use our applications to protect their phones and privacy from others.

## Related work

We tried to look up for any other nonvisual passcodes techniques but we could not find any. This means that there are not any nonvisual passcodes techniques for blind people have been studied. Instead, we could find some of studies on nonvisual input and output methods on smartphone. The main input and output methods of current smartphone are based on the phone screen, touching the phone screen as the input method and displaying on the phone screen as the output method. However, input and output methods of smartphone should be vision-free to become

<sup>1</sup> [http://www.askdavetaylor.com/enable\\_security\\_password\\_passcode\\_apple\\_iphone.html](http://www.askdavetaylor.com/enable_security_password_passcode_apple_iphone.html)

<sup>2</sup> <http://www.theawesomesite.com/android/how-to-retrieve-android-passcode.html>

accessible to blind people. Many different no vision requiring input and output methods on smartphone have been studied and developed and we could have some ideas in design of our applications from these related studies.

### Input function

Input functions are mainly used for searching and selecting buttons or icons on phone screen. One alternative way would be using smartphone Voiceover<sup>3</sup> function. Not only it does not require vision, but also Voiceover function is easy and fast enough to make an input by saving time in looking for buttons. Another way is making gestures on touch screen. Blind people can make gestures on touch screen as much as sighted people. On some gestures like flicking and multi-touching, blind people feel even easier than the sighted [1]. One example of finger gestures on touch screen is making braille characters on touch screen by fingers [2]. Since many blind people know braille and braille characters can be typed easily with fingers, it is really good alternative input method for the blind users. These techniques help them to make commands on the phone without vision.

### Output function

Sighted people see the phone screen and can recognize the status of phone. However, this is impossible for blind people. Same as input functions, many output functions working without vision have been developed. Most general examples are auditory [3] and vibration feedback. Auditory feedback tells the user any changes in the status of phone. Vibration feedback can be used to notify the user on special cases that it substitutes showing the alert message on the screen. With these methods, blind people can be notified any output of the phone.

### Nonvisual Passcodes

With given smartphone hardware which is not blind-friendly, we tried to figure out best input and output methods for our applications. Major techniques we implemented in our applications are hand gestures on touch screen as input function and auditory and vibration feedback as output functions. Since blind people have great searching ability with their fingers, we made all buttons in our application searchable with fingers by giving auditory feedback to help the user to find and select them. Not only the buttons on menu as shown in Figure 2, but also our applications give auditory feedback whenever the status of the application changes such as when the passcode is solved or not.



Figure 2. *BPass* and *GPass* main menu

Our first application is named *BPass*. *BPass* is the passcode that uses buttons at four corners on the smartphone. The main problem

in the existing lock screen is that it has buttons in the middle of the screen. Since these are not physical buttons, without vision, it is really difficult to find buttons in the middle of the screen. By putting the buttons on corners, users can track their hand by touching the frame of the phone. In *BPass*, users can make combinations of four buttons as their passcode. The number of combinations in the passcode is not limited. The phone recognizes the each of four buttons as four different digits from 1 through 4 as Table 1 shows and save the password as combination of four numbers.

We made some special implementations on *BPASS*. Not whole surface of smartphone is screen; there are extra areas outside of screen where cannot sense the gesture and can activate some other phone functions. In this reason, we blocked all other phone functions activated by touching outside of the screen, so the user's finger can freely move inside and outside of the screen. Also, we designed in a way that when the user touches the buttons, the phone will vibrate to notify the user whether the user touches the button or not. There are two cases that the user touches screen while using the passcode: when the user is searching for the starting button and when the user is actually drawing the passcode. To distinguish these two cases, we add a function to multi-touch case. The user can search the starting button of the passcode with a finger but the application does not recognize any touch at this point as passcode. Once the user finds the start button, tap the screen with a finger while holding the start button on screen with another finger to start input the passcode. Another problem we faced was that initially, our application thought inputting a passcode is ended when it does not sense any touch on the screen, but it was usual case that the finger can go outside of the phone while inputting the passcode, so the application thought the inputting passcode is done and stopped reading the passcode. To solve this problem, we also implemented multi-touch function to make the user able to give the command to the application when to finish reading the passcode.









Table 1. Buttons in *BPass* and their number representation

Button	Number Rep.
Top-Left Corner	1
Top-Right Corner	2
Bottom-Left Corner	3
Bottom-Right Corner	4

Our second application is called *GPass*. *GPass* is the passcode with combination of eight different hand gestures on touch screen. Like *BPass*, *GPass* does not limit the number of combination of gestures in passcode. Initially, *GPass* has more than eight gestures by making the direction of finger gesture matters. However, too much gestures and directions confuse the user and make it hard to use and eight gestures are still enough number to make a secure passcodes. Each gesture represents eight different numbers. So the password in *GPass* is a combination of eight numbers. Table 2 gives the description of each of eight gestures and its number representation. Unlike *BPass*, in *GPass*, the user has to tap with two fingers to finish the passcode because searching the button process is not necessary in *GPass*. However, the user has to learn these eight gestures before use, so we add tutorial mode in *GPass* where the user can learn and practice these gestures. In tutorial mode, it explains all eight gestures and the user can actually practice them; if the user makes a gesture, it tells which gesture the user has made, so the user can learn them. Same as in *BPass*, We blocked other buttons on the phone which can interrupt the user while inputting the passcode.

<sup>3</sup> <http://www.apple.com/accessibility/iphone/vision.html>

**Table 2. Gestures in *GPass* and their number representation**

Gesture	Description	Number Rep.
	One vertical line	1
	One horizontal line	2
	Two vertical lines with same direction	3
	Two vertical lines with opposite directions	4
	Two horizontal lines with same direction	5
	Two horizontal lines with opposite directions	6
	One vertical line and one finger staying	7
	One horizontal line and one finger staying	8

## User Study Results

We had 14 participants to test our applications and each participant tested both applications. All participants are sighted but while testing we made them to close their eyes. Table 3 shows the some statistics of test participants. While testing, we also monitored if any of factors (age, gender, or whether the current smartphone user or not) affects to the test result. It is possible that testing one application can affect to test the other application. For example, after testing *BPass*, participants could get used with using gestures on touch screen, and they would feel easier with using *GPass* or confused with gestures in *BPass*. To prevent this problem, we made 7 of them to test *BPass* first and other 7 to test *GPass* first. There are two main topics we tested: usability, and security.

**Table 3. Statistics of test participants**

Age	Mean: 26.5, Standard Deviation: 9.62
Gender	Male: 9, Female: 5
Smartphone User?	Yes: 9, No: 5

### Usability

We gave participants instructions on how to use our applications and let them try them until they understand how they work and become familiar of using them. Then, we asked each participant to make seven passcodes with different number of combination from one to seven and ask them to re-enter the passcode. We determined usability by checking the number of people who fail to solve the passcode in three trials - higher number of fails means more difficult to use.

We tested the passcode with length of 1 through 7 and nobody failed to solve the passcode within three trials. Some of them failed in first or second trials, mostly in *GPass*, but no participant failed to solve the passcode within three trials. This means that both *BPass* and *GPass* are easy to use. The reasons for failure in first or second trials with *BPass* are confusion between passcode or mistakenly touched wrong button but occurred rarely (2 out of 14 participants). For *GPass*, they confused between gestures or the phone misrecognizes the gestures and these occurred pretty often (6 out of 14 participants). We found that phone sometimes fails to distinguish between one finger and two fingers. We did not measure the exact time that took to solve the passcodes but, generally, *GPass* took less time than *BPass* in same length of

passwords. After testing both applications, we asked each participant about preference between them. All of them answered that they liked *GPass* more because it takes less time to solve and is easier to use. From comparing the results of participants, We did not find any effects from difference in participant's factors (age, gender, or whether the current smartphone user or not).

### Security

To test security of our passcodes, we let one participant make passwords and another participant watch the screen and the passwords. Then, we asked the participant who observed the password to try to solve it within 3 trials and if the participant solved the passcode within 3 trials, we assumed it as passed. We incremented the number of combinations until all participants failed to solve.

**Table 4. Security test result of *BPass***

Length of passcode	# of people failed
5	0
6	0
7	4
8	9
9	12
10	14
11	14

**Table 5. Security test result of *GPass***

Length of passcode	# of people failed
3	0
4	5
5	9
6	13
7	14
8	14

As Table 4 shows, for *BPass*, in number of combination in passcode from one to six, all participants could solve the passwords. Majority (>50% of participants) was unable to solve starting from the eight combinations. Table 5 shows that for *GPass*, in number of combination in passcode from one to three, all participants could solve and major people were unable to solve starting from five combinations. These results show that both applications require some reasonable number of combinations in passcode to keep security but *GPass* requires fewer numbers of combinations than *BPass*. Same as the easiness test, the factors (age, gender, or whether the current smartphone user or not) did not make any difference in test results.

From the results of tests, we figured out that most recommended number of combinations in passcode is eight for *BPass* and five or six for *GPass*. At length of eight, *BPass* can make 8748 different passcodes. At length of five, *GPass* can make 32768 different passcodes. Thus, at our recommending length of passcodes, guessing other's passcode is impossible and there is a little chance that different people have same passcodes but the users can remember their passcodes without confusion.

## Discussion

### Strength of *BPass*

*BPass* is easy to learn how to use. Since all users do to solve the passcode is connecting buttons on corners of the screen by a finger, any training before using it is not required. Also, it has little possibility that the wrong button will be inputted.

### **Weakness of BPass**

In *BPass*, the user has to find starting button and let the phone know when to start and end passcode in every time, so it takes little bit longer to solve the passcode than *GPass*. Also, *BPass* requires greater number of combinations in passcode to guarantee the security than *GPass*.

### **Strength of GPass**

*GPass* does not require effort to search for buttons. Users can directly make gestures on screen for passcode, so it takes shorter time to solve than *BPass*. *GPass* is also more secure than *BPass*. The user makes gestures quickly on *GPass* and some of the gestures use two fingers which make hard to catch both of them. These make the other people hard to guess the passcode even in small number of combinations in passcode.

### **Weakness of GPass**

Password in *GPass* is the combination of eight different gestures, so users need some training to learn these gestures and memorize them. Also, some devices have really sensitive touch screen, so they could confuse between different gestures. Thus, the user should be careful while making gestures.

### **Future Work**

Our applications need to be improved to workable on every smartphone devices or even on tablet PC. For *BPass*, in case of large phone screen or tablet PC, the distance between buttons is large, so it becomes difficult to drag from corner to corner. For these cases, we need to think about alternative algorithms over having buttons on corners of the screen. For *GPass*, we need to work on lowering the rate of misrecognition of gestures by phone. We can do it by making more differences between gestures. Since currently gestures in *GPass* have little differences, devices are confused between them. Current smartphone hardware is not blind-friendly, so there are limits on making best applications for blind people. If the hardware becomes more blind-friendly, it would be possible to make better passcodes for them.

## **Conclusion**

In this paper, we explored two passcode applications we developed. It was really tough to make user interface designs in this project because we have to think in perspective of blind people. Although they cannot see the screen, we found that there are other alternative ways to help their use of smartphones and many other people have studied on these subjects. We implemented gestures on touch screen and auditory and vibration feedback on our applications to make them sight-free. Through the tests, we proved that our applications are easy to use and secure. Although our applications still need to be improved more, we hope our applications give blind people more chance to use technology as much as the sighted people. Not only blind people, but also other disabled people should have equal chance to use technologies. And we think that we, software developers can help them to have this chance.

## **References**

- [1] Kane, S.K., Wobbrock, J.O., and Ladner, R.E.. Usable gestures for blind people: understanding preference and performance. In *Proceedings of the 2011 annual conference on Human factors in computing systems*. ACM, 2011.  
<http://userpages.umbc.edu/~skane/pubs/chi11-gestures.pdf>
- [2] Mascetti, S., Bernareggi, C., and Belotti, M.. 2011. TypeInBraille: a braille-based typing application for touchscreen devices. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '11)*. ACM, New York, NY, USA, 295-296.  
[http://delivery.acm.org/10.1145/2050000/2049614/p295-mascetti.pdf?ip=128.208.1.223&acc=ACTIVE%20SERVICE&CFID=70744079&CFTOKEN=83396516&\\_\\_acm\\_\\_=1331868249\\_14926573e1b34039f08a5e68c9f1249e](http://delivery.acm.org/10.1145/2050000/2049614/p295-mascetti.pdf?ip=128.208.1.223&acc=ACTIVE%20SERVICE&CFID=70744079&CFTOKEN=83396516&__acm__=1331868249_14926573e1b34039f08a5e68c9f1249e)
- [3] Li, K. A., Baudisch, P. and Hinckley, K. 2008. Blindsight: eyes-free access to mobile phones. In *Proc. CHI 2008* (Florence, Italy, 2008). ACM Press, 1389-1398.  
<http://research.microsoft.com/en-us/um/people/kenh/all-published-papers/blindsight-chi-2008.pdf>