Implicit Authentication for Mobile Devices Using Typing Behavior

Jonathan Gurary[1(✉)], Ye Zhu[1], Nahed Alnahash[2], and Huirong Fu[2]

[1] Cleveland State University, Cleveland, USA
j.gurary@vikes.csuohio.edu, y.zhu61@csuohio.edu
[2] Oakland University, Rochester, USA
{nalnahas,fu}@oakland.edu

Abstract. An attacker that compromises the unlock mechanism of a mobile device can fundamentally use the device as if it were their own, with access to a large portion of the user's sensitive data and communications. We propose a secondary implicit authentication scheme which monitors typing behavior to detect unauthorized use and lock down the mobile device. We build a basic implementation of our scheme on the Android operating system. Our user studies on the implementation show that we can achieve an accuracy of up to 97 % identifying one user out of a set of fifteen, with an FAR of $< 3\,\%$ and an FRR of $< .5\,\%$.

1 Introduction and Related Work

For most smartphones in use today, the sole defense against intrusion is the unlock mechanism that allows use of the device. An attacker who breaks the device's unlock authentication can gain access to the device. Access to a victim's cell phone allows an attacker to read sensitive communications, reset account passwords, and potentially access sensitive applications like those used for banking. A second level of real time authentication is desirable- even if an attacker gains access to the device, they will eventually be detected and locked out while attempting to use it. In order to remain viable, a real time authentication scheme must be accurate enough not to lock out legitimate users. In this paper, we propose an implicit authentication scheme based on soft keyboard typing behaviors which can identify users with a high degree of accuracy.

We choose to identify users with the touch screen because using the touch screen is a core behavior of any meaningful smartphone use. Physical biometrics such as gait [8] can be used as implicit authentication, however the attacker can merely refrain from walking while the device is powered on. Voice recognition schemes such as [2,4] can identify implicitly if the owner of the device is speaking nearby, but cannot help if the attacker stays silent. Interaction with the touch screen is required to place calls, write text messages, or access secure accounts and applications. A behavioral biometric that depends on touch behavior is difficult for the attacker to avoid if they wish to use the device.

Various authors have proposed schemes to authenticate users based on their touch screen gestures, touch patterns, and readings from on-device sensors during a touch [1,3,5–7,9–11]. Unlike previous research such as Feng et al. [6], which

T. Tryfonas (Ed.): HAS 2016, LNCS 9750, pp. 25–36, 2016.
DOI: 10.1007/978-3-319-39381-0_3

identifies users based on their touch screen gestures, we focus specifically on identifying users based on their typing patterns with the soft keyboard. Many users type small amounts of characters regularly on their smartphones, for example to send text messages, write emails, or dial phone numbers. An attacker that steals the device and attempts to use it for any of these tasks may be locked out, or they may discretely trigger an alert to a location schemes such as Apple's "Find my iPhone." Users may also type passwords on their mobile devices. An attacker that has compromised a user's account password[1] and attempts to enter that password into the device may be detected, locking the account.

Previous works such as Draffin et al. [5] have utilized typing behavior to authenticate users. Unlike previous works, our scheme utilizes all features that can be collected on modern smartphones, including device acceleration data, for an increased rate of accuracy. We also present several approaches for implementing touch classification, based on efficient statistical classifiers, and compare them using the same data. Our results show that by utilizing the large amount of data available on touch screen devices, we can achieve a 97 % rate of accuracy identifying users after only 15 touches.

In the next section, we present background and related work on keystroke dynamics and implicit mobile authentication using the touch screen. We continue by describing our implementation the data we collect in detail in Sect. 2. We present our user study and details about our classification approaches in Sect. 3, and we discuss our planned future work and extensions in Sect. 4. We conclude the paper in Sect. 5.

2 Scheme

We designed a basic implementation of the soft keyboard on the Android operating system. The built-in soft keyboard intentionally disallows recording of touch information to avoid various misuse such as keyloggers, and we found it was easier to build a basic keyboard than to attempt to override this security feature. A screenshot of our implementation is presented in Fig. 1. We use the built in Android class MotionEvent to collect data from touches on the keyboard buttons, and the SensorEvent class to collect accelerometer data. With these classes, we were able to record the following information about each touch:

Duration of Touch and Time Since Last Touch: The use of these values by themselves is sometimes called *keystroke dynamics*. We record the duration of touches and time between touches, in milliseconds. The duration of a touch is considered the time between the press and release of a button (eventtime-downtime in the MotionEvent class). The time since the last touch is considered the time from one press to the next (downtime-previous downtime in the MotionEvent class). For the first touch in every trace, the time since last touch is set to zero.

[1] We assume that the account utilizes trusted devices, and the attacker cannot simply enter the account information on another device.

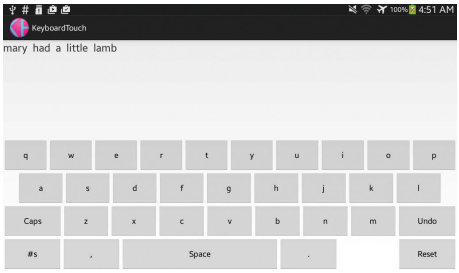


Fig. 1. A screenshot of our Android keyboard implementation.

Relative x and y Location of Press: The location of the center of the touch at the time the button was pressed, relative to the button, is recorded in pixel units. The top left corner of any button is (0,0) and the bottom right corner is the maximum, which varies by device.

Size of Touch on Press: The size of the touch is recorded on a scale from 0 to 1, where 1 is the maximum touch size the system will recognize, which varies by device. The system interprets all touches as a circle where size determines the radius of that circle. Size of a touch roughly correlates with finger size and touch pressure, which can be used to identify a user. The number of sizes supported by each device is not infinite; most devices support between 20 and 100 discrete touch sizes.

Magnitude of Acceleration on Press: The magnitude of acceleration is read in m/s^2 from the accelerometer, a sensor which the vast majority of devices on the market have today. Different devices update their sensors at different rates. We take the last known accelerometer reading before the press occurs, which may be several milliseconds before the touch itself.

Relative x and y Location of Release: The location of the center of the touch as the the button is let go. By taking the x and y locations of the press and release, we can determine how far, and in which direction, the user moves their finger during a touch. We do not use touch distance or direction in this paper directly because we are seeking to minimize computation requirements, though an intelligent classifier may utilize these features indirectly.

Size of Touch on Release: The size of the touch as the button is let go. We can infer pressure from the difference in sizes between press and release. If the press size is very large, and the release size is much smaller, then it is likely the user pushed their finger down hard on the device and increased the surface area that was making contact with the screen. We do not calculate pressure in this paper directly, though a classifier may indirectly take advantage of this relationship. Physical properties of the finger may also play into this feature, for example the amount an individual's finger yields when making contact with a hard surface.

Magnitude of Acceleration on Release: As with a press, the magnitude of acceleration on release is the last known accelerometer reading at the time

the user releases the button. If a touch is short enough and the device's sensor is slow to update, this value may be the same as the value for the press. The difference in acceleration between press and release can be used to infer how hard the device was touched or how steady the user's grip is. A large difference can indicate a hard touch which pushed the device down and and caused it to recoil back to its starting position. We make no effort to process acceleration in this paper, however our classifier may make such inferences indirectly.

Maximum, Minimum, and Average Acceleration during the Touch: We take as many readings as possible from the accelerometer between the press and release of a touch, recording the max, min, and average of the magnitudes of acceleration. Depending on the speed of the touch and the rate at which the accelerometer updates on that particular device, it is possible to obtain zero readings between the press and release, in which case we set all three values to some default value.

Although we are able to collect pressure from the MotionEvent class, we found that on most modern mobile devices pressure is either (1) set a default value which does not change or (2) scaled linearly with touch size. Capacitive touch screens like those found in most modern mobile devices are not able to sense the pressure of a touch directly. Many previous works that utilize the pressure reading from the MotionEvent class are actually utilizing touch size indirectly instead, and in works where both touch size and pressure were used, the use of pressure can be considered double-counting of touch size. Some manufacturers such as Apple and Huawei are currently developing phones which have touch pressure sensors, and some devices such as the Samsung Note feature a touch pen which can report pressure based on how hard the tip of the pen is pushed down.

Another feature frequently used in previous works is orientation, defined as the angle that the pointing device (e.g. a finger) makes with the screen. Orientation is also not reported by the MotionEvent class in many modern devices: it is either set to a single static value or one of two values depending on whether the device is held in portrait or landscape mode.

2.1 Future Implementation

We envision a scheme that collects touch data anonymously from a large pool of users and stores the anonymous data on some server. Periodically, touch data from random users is sent from the server to each client and the client generates a classifier for that data by combining it with their own data. Typing behavior from the user is analyzed by the classifier using one of the approaches described in Sect. 3. The user will have a high rate of accuracy matching with themselves, however the attacker will have a much higher chance of matching other users, with a very low chance to match the actual user several times consecutively. For example with 15 users, an attacker who has an equal chance to be behaviorally matched to any of those users has a $1/15 = 7\,\%$ chance to be authenticated as the user per attempt. If three consecutive failures cause a lockout, the attacker

has a $14/5^3 = 81\,\%$ chance to be locked out in the first three authentication attempts. We will also show that 15 touches or less can suffice for an authentication attempt. We will also demonstrate that we can attain high accuracy using a computationally simple classifier that can run in the background on a mobile device, with small sets of data that will require only minimal amounts of network use and device storage.

3 Experiment

3.1 Devices Used

We used a Galaxy Tab 3 tablet and a Google Nexus 4 smartphone for our experiments in order to ensure our results are consistent across different devices. The Galaxy Tab 3 has an 8 inch screen with a resolution of 1280 by 800, and the Nexus 4 has a 4.7 inch screen with a similar resolution of 1280 by 768.

There are several distinctions between the devices. We note that the Nexus 4 appears to report accelerometer readings more quickly, often enough that almost every touch reports a value for each of the acceleration features described in Sect. 2. In contrast, the accelerometer on the Tab 3 reports slowly. While it can collect information quickly enough for slow typists, for our faster typists as little as 5 % of touches report all the acceleration values described in Sect. 2. The Tab 3 reports the x and y location of touches to a precision of five decimal places, while the Nexus 4 uses only whole numbers. It is not clear to what digit the Tab 3's values are significant. Both devices report a large number of discrete touch sizes. The Tab 3 reports size on a scale from 0 to 1, while typical finger sizes on the Nexus 4 range from 9 to 11. We conclude that any implementation of our proposed authentication scheme will need to be device specific.

3.2 Experiment Setup

We recruited 15 volunteer participants to type the phrase "mary had a little lamb" on our tablet device and an additional 15 participants to type the phrase "maryhadalittlelamb" on our smartphone device. The space character is omitted for the second set of volunteers to determine if using the space character has a detrimental impact on identification accuracy. For example, we hypothesize that touches consecutive to space will consistently have a low time since last touch, since space is easy to find and reach on the keyboard, and this may make the time since last touch data point less useful in classification for those touches.

Participants typed the phrase a minimum of 20 times, though some participants chose to type the phrase up to 25 times. To ensure consistent acceleration data, participants were asked to sit in a stationary chair while typing, holding the device in landscape mode with their non-dominant hand and typing with their dominant hand. The particular grip participants used and the manner in which they typed were not specified, but participants could not use the hand holding the device to type or rest the device against any stationary surface. Participants were allowed to choose their stance and adjust it as they typed, for example they could lean forward or back in the chair.

3.3 Typographical Correction

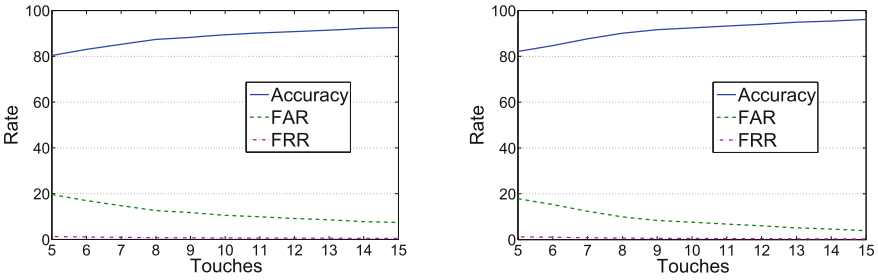
We employed some typological correction similar to Draffin's approach [5]. Typographical mistakes were treated as follows: (1) if more than three typographical errors were present, the trace was discarded, (2) if a character was typed incorrectly, e.g. "msry had a little lamb," the incorrect letter was treated as correct (treating "s" as an "a" in the example), (3) if a character was missing, e.g. "mry had a little lamb," the previous character would be duplicated, e.g. "mmry had a little lamb," and the typo ignored as in (2) and (4) if an extra character was typed, e.g. "masry had a little lamb," the extra character was simply removed. We hypothesize that some users make the same typographical mistakes consistently, and keeping these mistakes may actually help to identify them. For example, if the user attempts to press "a" and consistently hits "s," we expect the x-coordinate of the mistaken touch on the "s" key, after we correct it to an "a," to be more leftwards than for other users. We plan to analyze feasibility of identifying these mistakes in real time and the impact of keeping these mistakes in our future work. On average, users made approximately 10 typos in all of their 20 traces combined, though many of these typos are concentrated on specific users. Due to typo correction, some participants ended up with fewer than 20 traces, with a minimum of 15.

3.4 Classification and Analysis

We processed our data using computationally efficient classifiers- K nearest neighbors (KNN), binary decision tree, and naive Bayes. As smartphones have reached a high level of computational power, our hope is that these classifiers could run on the device itself in the future. In this paper we present only the binary decision tree results, as they were the most favorable for our experiments. We measure the success of our classification with (1) Accuracy: the percentage of authentication attempts from a user correctly matched to that user, (2) False Acceptance Rate (FAR): the percentage of authentication attempts matched to a user that do not belong to that user, and (3) False Rejection Rate (FRR): the percentage of authentication attempts from a user matched incorrectly to other users or rejected outright. We will define the meaning of an authentication attempt for each of our approaches later on.

3.5 Character Independent Classification

For this experiment, touches are grouped together and classified without considering the character being touched. We split touches evenly into testing and training sets at random, with approximately 200 touches per user in each set. All character information is stripped, that is an "a" is treated the same as a "b" or any other character. Training sets from all users are combined and fed to the classifier. Note that for participants typing on the Tab 3, this also means that the "space" character is treated the same as any other character, even though the space bar is significantly larger than other buttons and thus has a wider



(a) Galaxy Tab 3 (b) Nexus 4

Fig. 2. Touches vs Accuracy and FAR/FRR for character independent data.

range of position values. We hypothesize including space will reduce accuracy in the Tab 3's results.

An authentication attempt is begun by taking n touches for each user from the testing set at random and applying the classifier to each touch individually. The identity of the user is determined by taking the plurality of the n chosen touches. For example, n is equal to five touches and user a's trace contains two touches identified as user a, one as user b, one as user c, and one as user d. The authentication attempt is marked as successful for user a, even though the majority of the touches were attributed to other users, because the plurality of touches were identified as user a's. An authentication attempt using n touches is taken from users b, c, and d in the same manner. A tie does not authenticate any user and is automatically considered a false reject. To ensure consistency, we take 2000 samples from each user for each value of n, and the overall accuracy, FAR, and FRR are calculated by averaging the results for all users. Figure 2 demonstrates our results from $n = 5$ to $n = 15$.

Figure 2 shows we can achieve an acceptance rate of 93 % after 15 touches with an FAR of 7 % and an FRR of .5 % for the Tab 3 and an acceptance rate of 96 % with an FAR of 4 % and an FRR of .25 % for the Nexus 4. Thus a user can be authenticated after typing a short text message, using a training set that can easily be built in as little as two or three text messages. Our hypothesis that the space character will worsen results has held for this data set, though other factors such as screen size may also influence the different in metrics.

We theorize this approach could be used to dynamically monitor all typing on the device. Assuming a generous allowance of three incorrect authentication attempts before device lockout, a legitimate user will have a near zero chance of lockout ($7\,\%^3 = .0343\,\%, 4\,\%^3 = .0064\,\%$), while an attacker will face a substantial chance of lockout after only 45 touches. Because all characters are treated identically, classification data from other users utilizing the authentication application can easily be anonymously collected and distributed, allowing each user of the application to be compared against other anonymous users, potentially against different users for each authentication attempt. This further reduces the

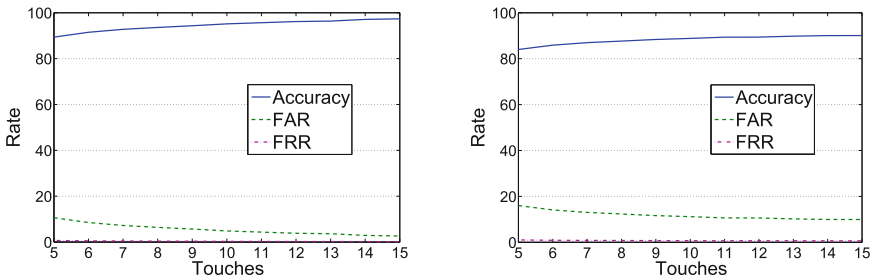
chance of a legitimate user being mismatched with another user with similar typing behavior but serves no advantage to an attacker.

3.6 Character Dependent Classification

For this experiment, each character receives its own separate classification. We classify the characters "a," "space," and "l" as these are the most common characters in our typed phrase. We split touches evenly into testing and training sets at random, with approximately 40 touches for "a" and "space" and 30 touches for "l" in each set. An authentication attempt for this experiment is defined the same way as in the previous experiment, with n touches taken at random from each user and a plurality-wins model for each authentication attempt. Once again we take 2000 samples from each user for each value of n.

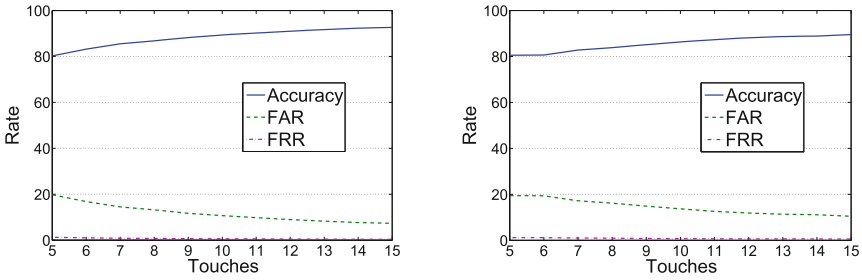
Figures 3, 4, and 5 show our results for each character from $n = 5$ to $n = 15$. We achieve an accuracy of 97 % after 15 touches of the letter "a," with an FAR of 2.6 % and an FRR of .2 % for the Tab 3 and an accuracy of 90 %, with an FAR of 10 % and an FRR of .7 % for the Nexus 4. For the letter "l," we achieve an accuracy of 92 % after 15 touches with an FAR of 7 % and an FRR of .4 % for the Tab 3 and an accuracy of 90 % with an FAR of 11 % and an FRR of .6 % for the Nexus 4. We note that for both characters, the Nexus 4 results are confounded by a single user who was consistently misidentified as one other user. Excluding this user puts the accuracy of the Nexus 4 above that of the Tab 3. Comparing against different users for each authentication attempt can reduce the probability of two users with very similar touch behavior getting confused with each other. Results for the "space" character on the Tab 3 are in line with other characters, achieving an accuracy of 95 % after 15 touches, with an FAR of 5 % and an FRR of .3 %.

This approach can be applied to frequent characters like vowels and space for reliable authentication with reduced overhead. As with the previous approach, the content and order of an individual's typed text do not matter, so anonymous classification data can easily be collected for different users. Comparing to different anonymous users for each authentication attempt can reduce the chances of



(a) Galaxy Tab 3 (b) Nexus 4

Fig. 3. Touches vs Accuracy and FAR/FRR for the Character "a"



(a) Galaxy Tab 3 (b) Nexus 4

Fig. 4. Touches vs Accuracy and FAR/FRR for the Character "l"

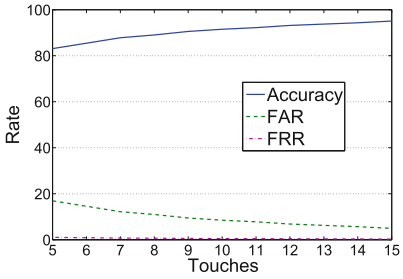
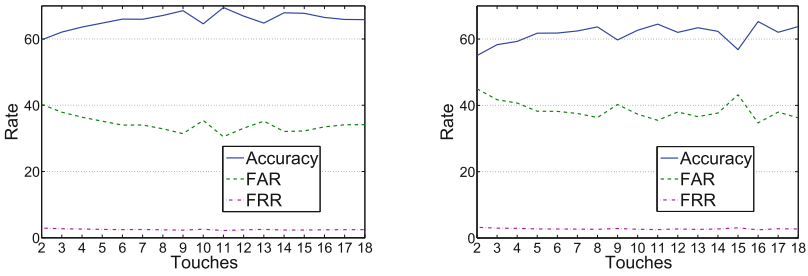


Fig. 5. Touches vs Accuracy and FAR/FRR for the Character "Space"

a consistent misidentification such as the one we encountered with the Nexus 4. While the success metrics for this approach are similar to the previous approach, applying classification only to popular characters can reduce the processing and memory overhead of the scheme.

3.7 Order Dependent

In our final approach, we consider how a user's typing behavior may change between different characters. In other words, a user may type the character "a" in a different way if "m" precedes it rather than "h," and this logic can further be extended to groups of 3, 4, or more characters. We believe this approach can be used for additional security on static text such as passwords. We keep the order of all touches and group them into pairs, threes, fours, and so forth, where n is the size of the group. The number of available traces depends on the size of n, for example there are 13 possible ordered letter pairs in our 22 character long phrase, and each user has approximately 20 traces, for a total of $13 * 20 = 260$ traces per user. Although success metrics may be different for certain letter combinations, e.g. for "ma" the results may be worse than for "ar," we combine the results and take their average for the purpose of condensing our results. We randomly select 60 % of the traces for training and 40 % for testing.



(a) Galaxy Tab 3 (b) Nexus 4

Fig. 6. Touches vs Accuracy and FAR/FRR for multiple consecutive touches

An authentication attempt is considered a set of n correctly ordered characters. We merge data from consecutive characters into a single row of data for purposes of classification. The entire row, containing data for each character in the sequence, is classified individually, so the authentication attempt is based on a single decision in this approach. This would be the only practical approach for a scheme designed to supplement password entry, since the user will generally enter their password only once per session.

From Fig. 6, it is clear that the increase in acceptance rate is actually quite minimal by using more consecutive touches. On a password of five of more characters, an accuracy of approximately 65 % is possible. Collecting classification data for an approach such as this one may be problematic, since other users will need to type precisely the same text. This approach may be applied to the entry of phone numbers rather than passwords, since phone numbers are also static. The number would not need to identical as the authentication can be done in three parts based on the area code, first set of digits, and final set of digits. Collecting classification data for phone numbers would be easier as many users will enter, at least, the same area code.

4 Discussion

There are two significant issues that our scheme faces before it could be applied to commercial use.

First, a smartphone is not a stationary object, and users frequently use their smartphone in different places and positions. In our experiment, we placed all users in a similar stance. Though most participants shifted stances slightly during the experiment, they still remained in largely the same positions. Different stances, e.g. walking, sitting, or lying down, must be identified, each with their own corresponding classification, because typing behavior will likely be different for the same user between these stances. Additionally being present in a moving object, for example a car, will affect acceleration results and potentially disrupt classification. While the scheme may work for the user most of the time without

regard to the user's stance, intelligent stance and acceleration detection will be required to use the proposed scheme in all situations.

Second, user behavior can change in accordance with mood, injury, time of day, sleepiness, etc. We collect test and training data in the same session to simplify our experiment. In our future work, we plan to take several samples of from users at different times and on different days to verify that our scheme can maintain accuracy despite day to day behavioral changes.

As gyroscopes have now become more prolific, we plan to include gyroscope data in our future work. Various other sensors, such as those that detect touch pressure, may also become more popular and ultimately justify further study. We also plan to experiment with derived values, such as distance traveled (calculated as vector between the start and end point of the touch), and force of touch (calculated based on differences in acceleration during touch) to see if a significant improvement in accuracy can be obtained.

We plan to investigate the performance of our scheme on user generated text input rather than preassigned text. We believe that some degradation of performance may occur because users frequently pause to think about the text they are writing and thus alter the nature of timing data to measure thinking speed as opposed to typing speed. The degradation in performance may be counterbalanced by the differences in text. For the order-independent approaches in our experiment, having users type different text increases the chance that their typing behaviors are dissimilar, decreasing the chance that two users will be confused with each other. We expect that challenges like typo detection will be significantly more difficult to implement on user generated text.

We note that in the first two approaches, the time since last touch feature may be considered as noise, since no information about the last touch is known, and each touch is treated as independent. We attempted our experiments without the time since last touch and found that the results worsened. We hypothesize that the time since last touch is proportional to overall typing speed and helps the classifier to identify users.

The data used in our experiments required about 1.6 MB and 375 kB for the first and second approaches respectively. Simple fast zip compression can reduce the file size to 475 kB and 90 kB respectively, so the scheme can have negligible impact on network use and device storage.

Lastly we plan to develop a method for the automatic anonymizing and transferring of classification data between users, for the purpose of building unauthorized classification samples for each user frequently and on the fly.

5 Conclusion

In this paper, we proposed an implicit authentication scheme for mobile devices that relies on touch behavior. We presented three approaches for classifying touch behavior on a mobile device: character and order independent, character dependent and order dependent, and character and order dependent.The first two approaches yielded an accuracy of up to 97 % with only 15 touches, using

statistical classifiers that are computationally cheap enough for implementation directly on the mobile device. Our third approach can be used as additional security for password entry, with an accuracy of approximately 65 % at five or more consecutive characters.

References

1. Antal, M., Szabó, L.Z., László, I.: Keystroke dynamics on android platform. Procedia Technol. **19**, 820–826 (2015)
2. Brunelli, R., Falavigna, D.: Person identification using multiple cues. IEEE Trans. Pattern Anal. Mach. Intell. **17**(10), 955–966 (1995)
3. Buchoux, A., Clarke, N.L.: Deployment of keystroke analysis on a smartphone. In: Australian Information Security Management Conference, p. 48 (2008)
4. Campbell Jr., J.P.: Speaker recognition: a tutorial. Proc. IEEE **85**(9), 1437–1462 (1997)
5. Draffin, B., Zhu, J., Zhang, J.: KeySens: passive user authentication through micro-behavior modeling of soft keyboard interaction. In: Memmi, G., Blanke, U. (eds.) MobiCASE 2013. LNICST, vol. 130, pp. 184–201. Springer, Heidelberg (2014)
6. Feng, T., Liu, Z., Kwon, K.-A., Shi, W., Carbunar, B., Jiang, Y., Nguyen, N.K.: Continuous mobile authentication using touchscreen gestures. In: 2012 IEEE Conference on Technologies for Homeland Security (HST), pp. 451–456. IEEE (2012)
7. Frank, M., Biedert, R., Ma, E.-D., Martinovic, I., Song, D.: Touchalytics: on the applicability of touchscreen input as a behavioral biometric for continuous authentication. IEEE Trans. Inf. Forensics Secur. **8**(1), 136–148 (2013)
8. Gafurov, D., Helkala, K., Søndrol, T.: Biometric gait authentication using accelerometer sensor. J. Comput. **1**(7), 51–59 (2006)
9. Maiorana, E., Campisi, P., González-Carballo, N., Neri, A.: Keystroke dynamics authentication for mobile phones. In: Proceedings of the 2011 ACM Symposium on Applied Computing, pp. 21–26. ACM (2011)
10. Saevanee, H., Bhattarakosol, P.: Authenticating user using keystroke dynamics and finger pressure. In: 2009 6th IEEE Consumer Communications and Networking Conference, CCNC 2009, pp. 1–2. IEEE (2009)
11. Trojahn, M., Ortmeier, F.: Biometric authentication through a virtual keyboard for smartphones. Int. J. Comput. Sci. Inf. Technol. **4**(5), 1 (2012)