KeySens: Passive User Authentication through Micro-behavior Modeling of Soft Keyboard Interaction

Benjamin Draffin*, Jiang Zhu, and Joy Zhang

Department of Electrical and Computer Engineering
Carnegie Mellon University
Moffett Field, CA, USA
benjamin.p.draffin@vanderbilt.edu, {jiang.zhu,joy.zhang}@sv.cmu.edu

Abstract. Mobile devices have become almost ever-present in our daily lives and increasingly so in the professional workplace. Applications put company data, personal information and sensitive documents in the hands of busy nurses at hospitals, company employees on business trips and government workers at large conferences. Smartphones and tablets also not only store data on-device, but users are frequently authorized to access sensitive information in the cloud. Protecting the sensitivity of mobile devices yet not burdening users with complicated and cumbersome *active authentication* methods is of great importance to the security and convenience of mobile computing. In this paper, we propose a novel passive authentication method; we model the micro-behavior of mobile users' interaction with their devices' soft keyboard. We show that the way a user types—the specific location touched on each key, the drift from finger down to finger up, the force of touch, the area of press—reflects their unique physical and behavioral characteristics. We demonstrate that using these micro-behavior features without any contextual information, we can passively identify that a mobile device is being used by a non-authorized user within 5 keypresses 67.7% of the time. This comes with a False Acceptance Rate (FAR) of 32.3% and a False Rejection Rate (FRR) of only 4.6%. Our detection rate after 15 keypresses is 86% with a FAR of 14% and a FRR of only 2.2%.

Keywords: Keystroke Dynamics, User Authentication, Passive Authentication, Multi-factor Authentication, Continuous Authentication, Biometrics, Micro-behavior, Soft Keyboards, Mobile Security, Android.

1 Introduction

Imagine a nurse has been using a mobile tablet to access and record sensitive patient information (such as in Figure 1). She is suddenly called away for an urgent question and without realizing it she leaves the tablet on the table—unlocked

* This work was done while the first author interned at Carnegie Mellon University, Silicon Valley under the supervision of Jiang Zhu and Joy Zhang.

G. Memmi and U. Blanke (Eds.): MobiCASE 2013, LNICST 130, pp. 184–201, 2014.

and still in the medical application. A curious or even malicious bystander picks up the tablet and searches though medical histories; they get unfettered access to private, personal information with not so much as a warning. Consider also the case where the nurse *did* lock the tablet; this attentive bystander may have easily learned a short PIN just by watching, passing right through traditional security barriers [9]. Active authentication can help protect devices at rest, but mobile, dynamic environments need mechanisms to detect and stop these security breaches in real-time. *Passive*, *behavioral authentication* measures are needed to counter these threats. We envision an application, *KeySens*, that develops a model of a user's micro-behavior and can detect when the phone is in a different person's hands. This application could then limit access or prompt for additional authentication upon detection. Towards that end, we have developed a proof-of-concept application and analysis models that demonstrate that users *do* have distinctive typing micro-behavior.

Fig. 1. Mobile devices such as iPads have become very popular among professionals. In this example, a nurse working at a dermatology clinic uses an iPad to take photos of a patient's skin, add notes, fill in forms and upload directly to the cloud.

1.1 Authentication Techniques

Authentication is the process of confirming that something or someone is what they say they are. The processes for authentication are many and diverse, and they have existed in the computing world since the beginning. Over time, the security field has categorized these into three primary groups [1]:

Something you know e.g. a password, security question, or ID number
Something you have e.g. a security token, ID card, or trusted device
Something you are e.g. a physical or behavioral trait such as a fingerprint or a keystroke dynamics model [14]

Most of these categories rely on *active authentication*, requiring direct user attention and input. While an effective protection schema to restrict unauthorized

access, traditional active authentication procedures are of limited use on mobile platforms. Users switch between tasks rapidly and are authenticating with dozens of services every day. Pausing and asking for passwords each time gets tedious and frustrating; users may choose to switch to less secure services with lower barriers to entry.

Passive authentication procedures are needed—ones that allow for transparent, low-friction authentication for known users but can detect and block unknown ones. Micro-behavior metrics are seen as a promising avenue for tackling this challenge. They are inherent or *latent characteristics* of the user and as such, they are very hard to impersonate. It is extremely challenging to fake a fingerprint, a retina, a vein pattern, a facial structure, or a hand geometry [9]. It also turns out that it is challenging to fake a micro-behavior metric such as signature, voice timbre, or a user's keystroke dynamics [4]. Mobile phones, equipped with a variety of powerful sensors and empowered with fairly substantial computing resources, are a perfect domain for deploying behavioral analysis to bolster their current security mechanisms.

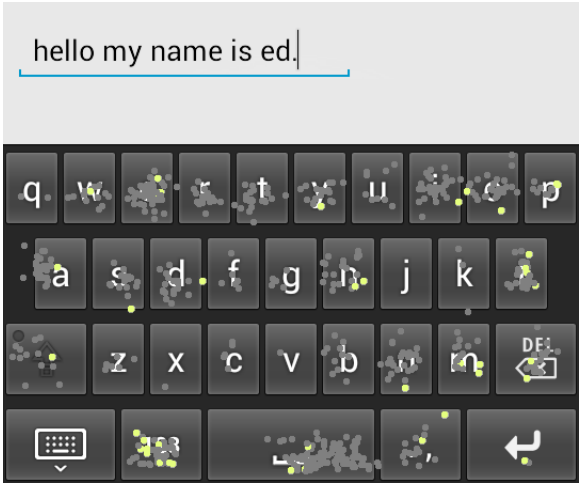

Fig. 2. Screenshot from an early version of the soft keyboard application comparing user typing patterns. In this example, a new user "Ed" presses different locations on keys (shown as yellow dots) than the normal user (grey dots) of this mobile device.

1.2 Contributions of This Research

This paper addresses how micro-behavioral metrics can be used to differentiate between users on mobile devices. This will be used for the development of a soft-keyboard application that can *passively authenticate* users in real-world environments. Keyboards and *keystroke dynamics* have been the subject of a wide body of prior research, looking to identify users based on *how* they type, not just *what* they type. We extend this research to the mobile platform and

leverage data available only on touchscreens to develop user models that work in real-world environments. A trial application has been developed that collects raw keypress information to be analyzed offline. Figure 2 is a screenshot from an early version of this application, visualizing how a new user ('Ed') may have different typing patterns than the typical user ('Mr. Grey Dots').

In contrast to traditional keyboard dynamics research, we analyze not only how rapidly a user types, but also a variety of soft-keyboard specific micro-behavior features. These include *where* on the key the user pressed, how much the user *drifts* over the course of a keypress, and the *orientation* of the phone.

Using this data, plus a variety of statistical tools, we can generate a *certainty score* of whether the user's phone is in a stranger's hands. This score, when combined with a larger application and decision engine, will be used in future work to block access to applications or ask for additional authentication information when an non-authorized user is detected. For this demonstration we leverage other users' keypress data to help train the micro-behavioral model, but we envision a system where all data is collected and stored on the phone and a model is trained offline separate from other user data.

1.3 Applications

With the potential to provide passive, transparent authentication on any mobile device with very little inconvenience, micro-behavior metrics have a wide range of target applications. Environments where sensitive data is accessed by busy, mobile people—one of the hardest areas to secure—can be further protected with these models. A few examples:

- Companies can help protect their data hosted on employees' phones (whether the hardware is company-owned or personally owned);
- Delivery persons or IT administrators who may need to place their tablets down to carry things can reduce risk of unauthorized access;
- Parents who lend their devices to children or friends can feel comfortable knowing they are protected against impersonating emails or other messages;
- Business travelers have reduced risk upon loss or theft of their device;
- Nurses who carry mobile devices to maintain patient records can be protected from prying eyes if they were to place the device down;

Additionally, with a content-agnostic and always aware view of typing patterns, these environments can be protected even in the event a non-authorized user learns the primary user's password through shoulder surfing, discovery of a note, or social engineering.

2 Related Work

While to our best knowledge there has not been prior published research on this particular topic, there has been a wide variety of high-quality, instructive

research on similar topics. These areas include desktop keystroke dynamics using traditional methods [1], mobile phone keypress dynamics using traditional methods [18] and key inference using side-channel sensors [13]. This research can focus on either *structured text* such as passwords/shared secrets or *dynamic text* that tracks keystroke patterns over the duration of a typing session. The testing environment in these studies could be either *controlled* or *uncontrolled*, describing whether the users are in a lab or on their personal machines. Additionally, studies can either focus on *authentication* of a single user or *identification* from among a pool of users [1]. They can also study a wide variety of features, depending on researcher interest and the capabilities of the target equipment.

2.1 Desktop Keystroke Dynamics

Desktop computer keystroke dynamics research has a long and rich history. A wide variety of extracted features, learning algorithms, success metrics and testing environments have been studied [1,14]. While more research has been focused on static text that can observe the patterns of password typing, there are a number of studies that focus on uncontrolled environments with dynamic text [2]. These studies, limited by the capabilities of physical keyboards, tend to gather information about typing latencies, cadences, or error rates. Some special keyboards can also provide pressure information, though this is not common in consumer-grade products. With our application, we leverage the additional data gathering power of a touch screen keyboard and can extend beyond features measurable on physical keyboards.

2.2 Mobile Phone Keypress Dynamics

Mobile phones have been getting increased attention over the last few years as security threats develop and risks are further revealed. The access to additional data from their touchscreens defines a research environment with both added benefits (e.g. more data) and added challenges (e.g. greater mobility, more dynamic environments). A number of studies have addressed keypress dynamics on mobile phones, though many use similar feature sets as on desktops [11]. Some, however, including a quite successful one on PINs by Zahid et al., analyze additional features such as the difference in *digraph time* between adjacent and non-adjacent keys [18]. Our work builds on this existing work by focusing on the mobile keypress features not possible on traditional desktops.

2.3 Mobile Phone Side-Channel Inference

Another growing area of research is the ability to leverage the other sensors on mobile phones (notably accelerometers and gyroscopes) to infer keypresses

without direct access to keypress data. By measuring the changes in orientation [3], accelerometer readings [13,3], and/or gyroscopes [3], applications with very few special privileges [15] can provide attackers insight into passwords or PINs. Applications posing as games or with other innocuous guises could tap into these sensor channels in the background and infer keypresses. We have internally discussed, though not examined, how a gaming application could generate its own training data through in-game menus. While we focus on the direct analysis of touches, we leverage this area of research to guide the processing of our own orientation data.

2.4 Other Features Used to Aid Authentication

The many sensors and input methods of mobile devices may be used to develop other micro-behavior metrics for authentication. Work on using accelerometer patterns to detect anomalous behavior was done by Zhu et. al. to success in their application *SenSec* [20]. Use of higher-level patterns from call /SMS frequency, ratios of known to unknown numbers, GPS locations and browsing history have been studied by Shi et.al [17]. Additionally, swiping patterns can also be used to differentiate between people, as shown to be a very accurate predictor in *Touchalytics* [6].

3 Microbehavior Modeling of Soft Keyboard Interaction

3.1 Soft Keyboard Interaction

Soft keyboards are a relatively new form of *input method editor* (IME). Many smartphones ship with no physical keyboard, instead opting for keyboards controlled by the touchscreen (e.g. Figure 2). An image of a keyboard appears on screen and users tap the keys they would like to send. Challenges arise, however, because of the lack of physical feedback from buttons. Improper estimates of finger tap locations due to the small keys getting completely covered by fingers in addition to finger drift due to the smooth surface make these keyboards sometimes less appealing to use. However, many of these disadvantages help enable researchers to track these touch patterns over time to build a user profile. We took advantage of these features to build an trial *input method editor* (IME) to gather example user data. This application (implementing a custom KeyboardView for raw touchscreen access) records absolute pointer positioning and size/pressure data for each finger or stylus. On most Android phones, there can be up to 4 or 5 touchscreen events per keypress, giving us an abundance of data with which to work.

3.2 User Keypress Variations

While typing, users have a variety of different typing rhythms and the physiological traits of their hands, joints and fingertips ensure that no two users type

exactly alike. Alongside traditional desktop-like features such as experience level and posture, soft-keyboard typing seems to be influenced by hand size, finger length, fingertip size, muscle development, posture and position, one-handed or two-handed typing, orientation of the phone, user tiredness, coldness of fingers, focus of the user (mobile users are often partially engaged in other activities), and whether the user is walking, standing, sitting, lying down, or riding in a car, bus or subway. Additional influences are size/shape of the phone, screen, and external phone cases. Combined with technology literacy and familiarity with the typing application, real-world analysis of mobile keypress dynamics reveals itself as a significant challenge. We have considered these influencing factors and analyzed the available features on Android phones and have designed a feature set (Section 3.3) to match appropriately.

3.3 Feature Selection

After analyzing the prior research on keyboards as well as the smartphone-specific information described above, we developed our set of features to capture and analyze. These will be used to help train our model and be will analyzed for patterns between users. Later in the paper are a few visualizations of how these features can vary between users.

Location pressed on key is our primary micro-behavior metric and is recorded per key as an ordered pair $(X_{\text{offset}}, Y_{\text{offset}})$, expressed from that key's center.

Length of press from finger down to finger lift, a traditional metric, has much greater variation on mobile phones, but does improve model performance.

A user's *force of press* or *pressure* is available as a unitless value between 0 and 1 on many Android devices [16]. This allows for tracking of metrics such as distributions of *maximum touch pressure*, but is complicated by inconsistent scales between device models.

Similarly, the user's *size of touched area* is also available as a unitless value between 0 and 1 [16].

We can also analyze variability in *size* and *pressure* information within a single keypress. This allows analysis of *pressure and size dynamics* for the user.

Another feature, *drift*, records how much the user's finger moves between pressing down and lifting up and at what angle they slid.

Additionally, with easily available sensor data from accelerometers and gyroscopes, information about *orientation* (where the phone faces while held) could help improve comparisons and anomaly detection. This proved to be challenging to correlate between handsets, but the addition of a calibration session could enable successful consideration of this feature.

4 Application Considerations

4.1 Data Privacy and Security

By collecting so much data about each touch event on the keyboard, we are able to compare a wide variety of features between users and develop an improved

model for user authentication. However, much of this information is highly sensitive or confidential. We cannot expect users to willingly use a keylogger on their personal devices and we certainly do not want to expose scores and scores of key sequences in the unfortunate event of server compromise. Given that we do comparisons key-by-key, it was necessary to find an obfuscation strategy that allows us to know which key was pressed, but cannot let someone find out *when* it was pressed. Thus, we remove timestamp information from the keypresses and order them by a cryptographically secure random number generator (Java's SecureRandom class) so that the original sequences are not reorder-able. This, combined with block-level encryption and large-batch HTTPS transfers to our server help protect user data while allowing us access to the information we need.

4.2 Power Consumption

Mobile devices are highly sensitive to power consumption considerations. Users expect that applications not only improve their mobile experience, but do so with limited impact on battery life [7]. It was observed from our users that the soft keyboard application was consuming less than 4 percent of the phone's battery life. For most, it did not even show up in the listing (alongside power hogs like Maps, Music). Further analysis of current draw is needed, but power consumption is markedly low. This will likely increase, however, once the behavioral model computation is done directly on the handset.

4.3 Development Concerns and Programming Challenges

While developing and testing an Android keyboard, a variety of challenges presented themselves:

- Multiple fingers need to be tracked independently. Android represents multitouch gestures in combined Objects, but fingers in the same gesture must be recorded separately using independent identifiers.
- Phones have different size screens; press locations and offsets need to be scaled.
- The Android operating system has depreciated the high-level construct 'Orientation' as of API level 8, and correlating phone orientation information is very challenging between devices.
- When users intend to press a key—for example 'k'—and accidentally press 'j', we pick up misleading data points about where they tend to press on the 'j' key and miss out on a valuable data point about 'k'. Almost all keys have outliers of this type due to the occlusion of the actual touch location by the rest of the finger [8]. We monitor for users' corrections and modify the 'intended key' appropriately. This enables us to collect typical press information even upon mistakes, and we can analyze which keys are frequently mistakenly pressed.

5 Experimental Design

5.1 User Recruitment

A short recruitment drive resulted in a group of 13 trial users. The users were mostly technology literate students, as well as a few professors. To ensure there were a sufficient number of test users and to encourage natural typing behaviors, our application was designed to closely mirror the features of other android keyboards. We feel that spending time on the user experience side of application development was a worthwhile investment for increased user retention and positive reactions to the typing environment. If users have to spend a long time adjusting to the new system, their early data can be misleading or even counterproductive when developing a training model.

5.2 Data Collection

A three week long collection period resulted the a group of 13 active users contributing a total of roughly 86,000 keypresses (with highly variable contributions from each user) and about 430,000 touch data points. This gives some indication about how much data can be collected in a short period of time, especially when gathering keypress information in all contexts, not just from passwords or controlled phrases.

5.3 Training System Design

When designing learning algorithms, it is very important to ensure good training practices are followed [12]. Model training, validation and testing are important parts of the process and should be considered carefully. When analyzing, user data is split into five sections, the first four randomly sampled from the first two weeks of collection. The final set of testing data is from at least 3 days after the rest (to test in user environments independent from training data). Our results are generated from the 'final testing data.'

1. Training data for primary generative or discriminant models (50% $\simeq$ 3000 keypresses from user, 2000 from each of 3 other random users)
2. Cross validation for primary model (15%)
3. Testing data for model and training data for key-frequency scaling (10%)
4. Cross-validation for key-frequency scaling (10%)
5. Final testing data (15%)

6 Modeling Micro-behavior of Users' Keyboard Interaction

6.1 Discriminant Model

We developed two models for comparison testing between users—each suited for different environments. The method of analysis we used for the results in this

paper is offline supervised learning trained with data from other users. While not as scalable as a purely generative version, this has some additional advantages—better non-authorized user recognition, better battery life by reducing on-phone computation, and more securely stored behavioral data (on server rather than directly on phone) [19]. This technique trains small neural networks for each key, using samples of other users' behavior as 'non-authorized user' training examples [5]. A high-level diagram explaining this technique can be seen in Figure 3 and the following paragraph contains a detailed explanation.

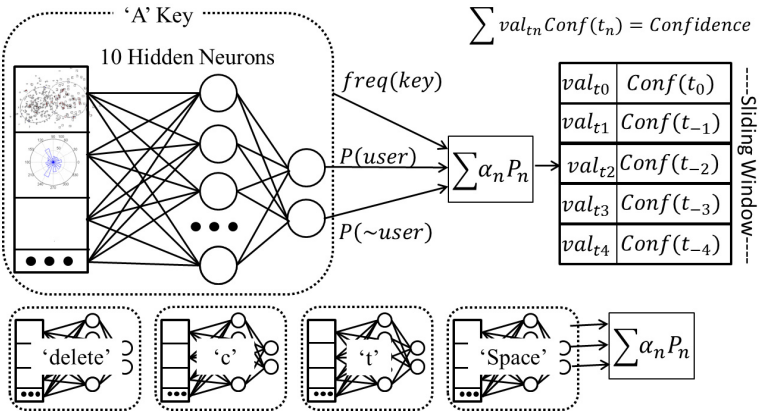

Fig. 3. A high-level diagram of how the discriminant algorithm is designed. Per-key neural networks generate confidence scores weighted against training size. These are aggregated into a combined score for a 5-key sliding window to generate likelihood of non-authorized user.

This model leverages the multi-user data collection environment and takes advantage of the high number of training keypresses. While more computationally expensive, this facilitates much higher recognition rates and fewer errors. Each key with a reasonable amount of training data (we set threshold at 15 keypresses) is trained with its own two-layer, feed-forward neural network. Scaled conjugate gradient back-propagation is the learning method [5]. We used 10 neurons for the one hidden layer to ensure there is sufficient power in the analysis. The performance function is the *mean squared error*, but another performance function checks after training to ensure the network is not ill-fitted. This may ask networks to retrain if they settle in poor local minima (e.g. classifying everything as the regular user). The output from these neural networks are then weighted by how many keypresses were used to train that key. The recent confidence scores are averaged using a simple mean and this is compared against a threshold. The threshold (and the per-key weighting) is set by using a regularized logistic regression algorithm [5]. The cost function for the algorithm is shown in Equation 1. To improve scalability and to allow for non-networked computation, this model will later be replaced with a single-user generative model (described below).

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} \left(-\log h_\theta(x^{(i)}) \right) + (1 - y^{(i)}) \left(-\log(1 - h_\theta(x^{(i)})) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 \tag{1}$$

6.2 Generative Model

Our intention is to further develop our generative model that can detect anomalies without network access or comparison to other users. A generative model is needed for this approach. For a single user's features, historical distributions can be determined and models (Gaussian or otherwise) developed [10]. Then, upon new data input, the probability that each feature has come from the user is calculated. Aggregated for all of the features, these probabilities give a per-key confidence score. Additionally, as per an evaluation by Killourhy and Maxion, an *outlier count* feature was added that tallies the number of recent outliers (shown to be effective at detecting anomalies on desktops) [10]. In a similar fashion to the discriminant model, number of historical keypresses at that key serves as the weight relative to other recent keypresses. See Figure 4 for a visual explanation. This weighting is done through the same logistic regression as described in the 'discriminant model' section but is expanded to include the input features processed by that model's neural network [5].

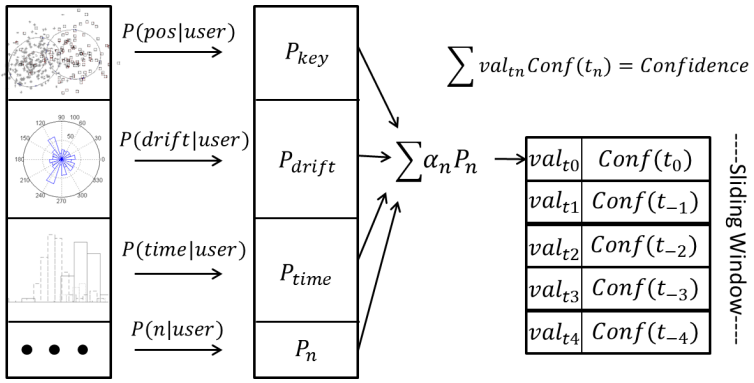

Fig. 4. A high-level diagram of how the generative algorithm is designed. It calculates feature probabilities individually, aggregates them to a confidence score for a single key press and uses a 5-key sliding window to generate likelihood of non-authorized user.

6.3 Feature Distributions

Described below are a few of the key metrics we analyzed for aptness of discrimination. The visualizations are generated from similarly-sized samples representing a variety of users and have been selected to be representative of typical

variation of micro-behavior. Note again that we do not have contextual information such as movement speed, time of day, or number of hands typing, so this data is not filtered in that way.

Variability in Keypress Location was predicted to be the best differentiator between users, and it turned out to be correct. Figure 5 displays how there are particular patterns that develop for users on specific keys. 'User 1' is almost always in the bottom right, while 'User 2' is usually right around the vertical center of the key but with wide horizontal variability. Observe also that there are two area of concentration for 'User 2.' Based on some contextual knowledge about that user's behavior we found that the left concentration is from the left finger, while the right concentration is from the right finger. Figure 6 compares 5 different users' press locations on a single key, holding 'User 1' constant for reference.

With the key press location data, we applied a *bivariate Gaussian distribution* for each key. The formula for bivariate covariance is in Equation 2 [5].

$$cov = \frac{1}{n}\sum_{i=1}^{n}(w_i - \bar{w})(w_i - \bar{w})', \tag{2}$$

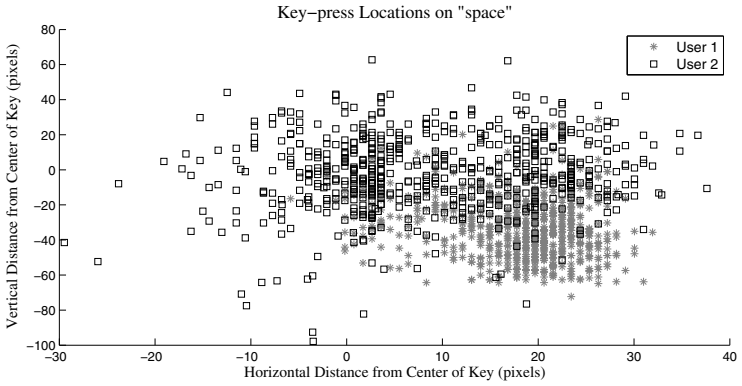

Fig. 5. Two-User Comparison of Keypress Locations on 'spacebar'. Observe how users tend to clump, and how 'User 2' presses in seemingly distinct areas with their left thumb versus right thumb.

Keypress Length, a common metric of authentication on desktops, turned out to be ineffective at discriminating users on mobile devices. While experienced desktop users tend to type faster than inexperienced ones [2], mobile users type in a much wider range of circumstances and at a much wider range of rates. Keypress length does, however, divide users into three general categories: fast, medium, and slow typers. If a user is very consistent, a deviation could reveal an unauthorized user.

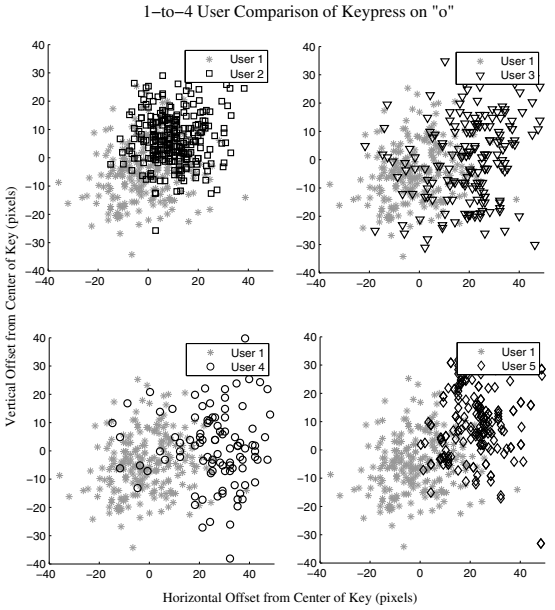

Fig. 6. Quad-comparison of Keypress Locations on 'o'. 'User 1' and the axes are static. This shows how different users have (sometimes widely) varying spreads and centers.

Drift was found to have different angle distributions depending on the user. Due to touchscreen noise, drift is only counted if a finger's last contact is more than 4 pixels from point of first contact. The numerical cutoff is referred to as the *drift threshold*. See Figure 7 for a visualization of drift angle distributions where the origin is normalized location of first contact.

The data for *drift* is by no means normally distributed, but tends to clump into a number of distinct areas for particular users. By grouping angles into a finite number of buckets (akin to the rose histograms below), we can calculate the historical probability of a particular drift direction against which to compare new data. Our formula for such analysis is below:

$$P(drift|user;\theta) = \alpha_{drift} * P(anyDrift|user) * P(\theta|user) \tag{3}$$

Pressure, Size and Orientation revealed themselves to be rather challenging to manage features. For an individual user it is possible to measure the average and max *pressure* of the finger. It is, however, very challenging to correlate this data between users on different styles of handset; different screen technologies report pressure using different scales.

Additionally, the *size* of a user's finger on the screen is frequently not possible to compare between different phone styles (but can be analyzed for anomalies

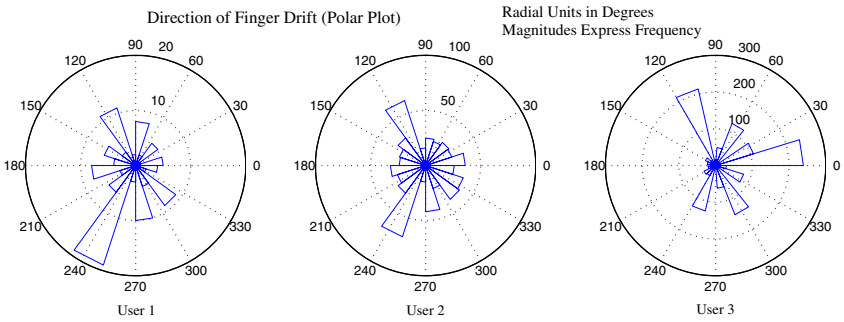

Fig. 7. Comparison of finger *drifts* between users. The center of each circle is the normalized location where the user first pressed down. The angles describe which direction the user drifted (but not how far). The ray lengths are the frequency of drifts in that direction.

for one user). The size is a unit-less value and can vary widely from device to device. Further research on multiple users using identical handsets is required.

Orientation suffers from a similar problem, though it should be possible to ask testing users for a calibration session upon installation. This may allow regularization between devices.

7 User Identification Results

7.1 Success and Error Metrics

When evaluating the success of a model or a process, it is important to define the metrics used. Our primary objective is to identify 'non-authorized users' quickly, accurately, and repeatably. This is essential in creating a functional authentication system. Our other main objective is to minimize how often the primary user is flagged as another user—an inconvenience. For these metrics we define *Detection Rate* as the frequency of successfully detecting 'non-authorized users', *False Rejection Rate* (FAR) as the frequency of flagging the primary user as 'non-authorized' and the *False Acceptance Rate* (FAR) as the frequency of failing to flag a 'non-authorized user' as such [12].

7.2 Attack Detection

Our discriminant algorithm trained on multiple users performed well in testing, detecting an median of 67.7% of simulated 'non-authorized users' within 5 keypresses. Our False Acceptance Rate (FAR) was 32.3% and we had a False Rejection Rate (FRR) of only 4.6%. For longer input sessions of 15 keypresses, our detection rate rose to 86.0% with a FAR of 14.0% and a FRR of only 2.2%. Models were trained with 3000 keypresses from the 'primary user' and 2000 from each of 3 other users. It was then tested against 550 'primary user' keypresses

and 500 'non-authorized user' keypresses from a variety of other users. A few keypresses from each user were ignored due to lack of training data for those keys (a potential concern for analyzing symbol-heavy passwords). The 'non-authorized users' in the testing sets were not used for training data. The test data from the primary trainee was from at least 3 days after the training data to ensure independent (though not assuredly distinct) environments. The performance matrix and receiver operating characteristics (ROC) curves (for a number of different users) are in Figure 8.

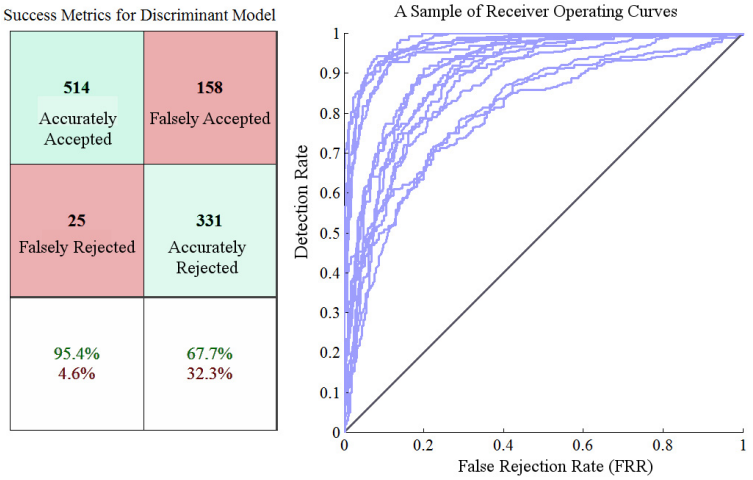

Fig. 8. Testing performance and receiver operating characteristics for discriminant model. 67.7% of simulated 'non-authorized users' were caught within a 5 keypress sliding window with a False Acceptance Rate of 32.3% and a False Rejection Rate of only 4.6%. The ROC curves represent a subset of all tests indicating variability of detection rates between users.

As seen in the ROC curves (a subset of all tests), there is a sizable spread to the recognition rates between users. This needs to be further explored, however it indicates that some users are easier to tell apart than others. Running a model trained on 'User 1' with two different sets of example 'non-authorized users' can generate very different detection rates.

Our results do suffer from a significant hole, however. Because we strip timestamp information from the data before logging, we cannot recreate exact strings to use as testing data. We counter this as best we can by creating each test string with data from a single log file (which typically contain only a few minutes of typing data). Further testing needs to be done on data known to be in ordered strings, however actual keylogging will only be acceptable in more limited environments.

8 Discussion and Next Steps

The next step of this research is to develop an on-device analysis tool that enables a live, learning model of users' behavior and the ability to flag suspicious activity (preferably without requiring network access). This could enable rapid recognition of suspicious activity without network lag, and also improve scalability without requiring additional server resources. The computational load would likely fairly substantial, but because it could be run only while the user is typing, it should not burden the phone's resting battery life.

Additional studies will be necessary to better understand a user's typing dynamics over time and in variable situations. Additional contextual information about phone usage could be used to improve accuracy and the robustness of the algorithm. Moreover, testing environments where users are using unfamiliar phones for the first time will enable direct comparison between unauthorized and primary users and may better reflect attack scenarios.

Collaboration between research groups can also be a highly fertile ground for future work. The groups working on user interface interaction characterization or side-channel keypress inference could be excellent teammates combining expertise on touch information and motion sensor readings. Also, groups looking into the trend towards swipe-based keyboards could leverage some of these techniques to develop similar models. Perhaps individual users' swiping gestures are also unique enough for micro-behavior authentication.

With this micro-behavior information and potential for improved phone security, these kinds of applications should be of great interest to businesses trying to improve "Bring Your Own Device" security policies. A stipulation of private phone usage could be a requirement to use a behavioral-modeling keyboard. Company provided phones could also integrate these features into a semi-customized OS that could analyze any user-level input method.

8.1 Deploying Behavioral Modeling to Secured Mobile Devices

After examining a variety of Android operating system functions and features, it became clear that there are a few security challenges the system still faces. Were behavior-modeling keyboards to be integrated into a public or private Android distribution, our team recommends a few enhancements that could improve device security:

- Weaken the potential for side-channel attacks by disabling sensors while entering text into password fields. This could substantially reduce the chance of reconstructing strings via accelerometer or gyroscope data.
- Require use of the default Android keyboard (or a manufacturer-vetted application) during password or sensitive text entry. Risk from malicious keyboards is likely to grow in the future and users may not carefully read privacy policies or permissions for their applications [15].

9 Conclusion and Future Work

Protecting user devices in mobile, dynamic environments is of essential importance in the academic, business, and personal worlds. We have taken a dramatic step towards developing *passive keyboard authentication* on smartphones. We demonstrated that using these micro-behavior features, we can *passively detect* that a mobile device is being used by a 'non-authorized user' within 5 keypresses 67.7% of the time. This comes with a False Acceptance Rate of 32.3% and a False Rejection Rate of only 4.6%. For longer input sessions of 15 keypresses, our detection rate rose to 86.0% with a False Acceptance Rate of 14.0% and a False Rejection Rate of only 2.2%.

Our long-term objective is to integrate a fully developed *KeySens* application with our team's larger, more inclusive sensor suite, *SenSec*, to provide multi-dimensional pattern recognition features [20]. Accelerometers, gyroscopes, awareness of opened applications, and other such features can help demonstrate the feasibility of an always-aware authentication structure. This larger project also has a technique for completely blocking access to applications and requesting an active authentication if the *confidence score* drops too low. We hope that our work will inspire further research into this subject. Passwords and PINs alone cannot protect mobile users. By deploying them in conjunction with micro-behavior metrics and other authentication techniques, mobile devices can become an ever-safer place in the computing world.

Acknowledgments. This work is supported in part by CyLab at Carnegie Mellon under grants from the Northrop Grumman Cybersecurity Research Consortium and by Cisco under the research award for "Privacy Preserved Personal Big Data Analytics through Fog Computing".

References

1. Banerjee, S.P., Woodard, D.L.: Biometric authentication and identification using keystroke dynamics: A survey. Journal of Pattern Recognition Research (2012)
2. Bergadano, F., Gunetti, D., Picardi, C.: User authentication through keystroke dynamics. ACM Trans. Inf. Syst. Secur. 5(4), 367–397 (2002)
3. Cai, L., Chen, H.: On the practicality of motion based keystroke inference attack. In: Katzenbeisser, S., Weippl, E., Camp, L.J., Volkamer, M., Reiter, M., Zhang, X. (eds.) Trust 2012. LNCS, vol. 7344, pp. 273–290. Springer, Heidelberg (2012)
4. Cherifi, F., Hemery, B., Giot, R., Pasquet, M., Rosenberger, C.: Performance evaluation of behavioral biometric systems. In: Behavioral Biometrics for Human Identification: Intelligent Applications, pp. 57–74. IGI Global (2010)
5. Duda, R.O., Hart, P.E., Stork, D.G.: Multi-layer neural networks. In: Pattern Classification, 2nd edn., vol. 2. John Wiley and Sons, Inc. (2001)
6. Frank, M., Biedert, R., Ma, E., Martinovic, I., Song, D.: Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. IEEE Transactions on Information Forensics and Security 8(1), 136–148 (2013)

7. Gordon, D., Czerny, J., Beigl, M.: Activity recognition for creatures of habit. In: Personal and Ubiquitous Computing, pp. 1–17 (2013)
8. Holleis, P., Huhtala, J., Häkkilä, J.: Studying applications for touch-enabled mobile phone keypads. In: Proceedings of the 2nd International Conference on Tangible and Embedded Interaction, TEI 2008, pp. 15–18. ACM, New York (2008)
9. Jain, A., Hong, L., Pankanti, S.: Biometric identification. Commun. ACM 43(2), 90–98 (2000)
10. Killourhy, K.S., Maxion, R.A.: Comparing anomaly-detection algorithms for keystroke dynamics. In: IEEE/IFIP International Conference on Dependable Systems Networks, DSN 2009, pp. 125–134 (2009)
11. 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, SAC 2011, pp. 21–26. ACM, New York (2011)
12. International Standards Organization. Biometric performance testing and reporting (2006)
13. Owusu, E., Han, J., Das, S., Perrig, A., Zhang, J.: Accessory: password inference using accelerometers on smartphones. In: Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications, HotMobile 2012, pp. 9:1–9:6. ACM, New York (2012)
14. Peacock, A., Ke, X., Wilkerson, M.: Typing patterns: a key to user identification. IEEE Security Privacy 2(5), 40–47 (2004)
15. Android Open Source Project. Android security overview
16. Android Open Source Project. Touch devices
17. Shi, E., Niu, Y., Jakobsson, M., Chow, R.: Implicit authentication through learning user behavior. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 99–113. Springer, Heidelberg (2011)
18. Zahid, S., Shahzad, M., Khayam, S.A., Farooq, M.: Keystroke-based user identification on smart phones. In: Kirda, E., Jha, S., Balzarotti, D. (eds.) RAID 2009. LNCS, vol. 5758, pp. 224–243. Springer, Heidelberg (2009)
19. Zhu, J., Hu, H., Hu, S., Wu, P., Zhang, J.Y.: Mobile behaviometrics: Models and applications. In: Proceedings of the Second IEEE/CIC International Conference on Communications in China (ICCC), Xi'An, China, August 12-14 (2013)
20. Zhu, J., Wu, P., Wang, X., Perrig, A., Hong, J., Zhang, J.Y.: Sensec: Mobile application security through passive sensing. In: Proceedings of International Conference on Computing, Networking and Communications (ICNC 2013), San Diego, CA, USA, January 28-31 (2013)