Passive User Identification Using Sequential Analysis of Proximity Information in Touchscreen Usage Patterns

Vadim Zaliva*, William Melicher†, Shayan Saha‡, Joy Zhang§
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA, USA
Email: *vzaliva@cmu.edu, †billy@cmu.edu, ‡shayan.saha@sv.cmu.edu, §joy.zhang@sv.cmu.edu

***Abstract*—Modern touch screen sensors are capable of detecting and reporting finger presence not only upon contact but also as the finger is approaching the screen. This gives us a wealth of additional information, which to the best of our knowledge, has never been analyzed before. Using these new sensor capabilities, we can see exactly how a user performs gestures starting from the finger's approach through the actual touching of the screen.**

We decode proximity data which we collect from the mobile phone sensor and extract finger "traces" from each user along with the contact area shapes, which we use to distinguish between the owner and one of the other users. To further improve the classifier's accuracy, we develop a sequential classification approach using a probability ratio test of artificial neural network outputs which makes a decision in minimal time based on predefined accuracy goals. The data not only allows discrimination between users but also detection of their dominant hand. These techniques could be used in many practical applications, such as passive user authentication or personalization.

Our experiments show that after just 5 touches, or in 12.6 seconds on average, we can correctly distinguish the primary user from any of 14 other known users using proximity data to model the finger's approach pattern.

I. Introduction

The rapid adoption of smart phones over the past 5 years has changed our communication habits. A large portion of our information access, generation, and sharing has shifted from desktop to mobile platforms. Consequently, more and more sensitive information, such as email and financial data, is kept on mobile devices. In addition, the small and portable nature of mobile devices makes them easy targets for attackers. Stolen devices and unauthorized use of mobile devices have become major concerns in information security.

Modern touch screen sensors are capable of detecting and reporting finger presence not only upon contact but also as the finger is approaching the screen. In this paper, we study the effectiveness of using proximity sensor data to model the finger approach behavior of a user, and we exploit this approach behavior for passive authentication.

Modern mobile phones equipped with proximity-sensitive touch screens include Samsung Galaxy S4 and Sony Xperia. We distinguish three phases of user interaction with the touch screen. An *approach* phase starts when the sensor first detects a finger prior to its physical contact with the screen. The sensor provides the finger distance from the screen and its position in screen coordinates. The *touch* phase commences when the finger actually touches the screen. At this moment, besides actual touch position, additional information about the shape of the finger's imprint becomes available. After the touch, the *tracking* phase records the finger position as long as it remains in contact with the screen.

We envision a scenario in which a background process continuously analyzes the user's finger approach and touch information in real-time. If it detects that the phone is being used by someone other than the owner or primary user, the process could lock the screen and require the appropriate password to unlock it. Under this scheme, a preemptive mandatory user authentication (entering a PIN when the phone is switched on) is replaced by non-intrusive background monitoring, which could arguably be more user-friendly.

II. Background and Related Work

Several works have used mobile sensor data to extract a model of users' behavior for authentication [5], [6]. Some works concentrate on authentication while others, like this work, concentrate on identifying a user from a pool [1], [12]. On desktop computers, this type of passive authentication has been studied using *keystroke dynamics* [7], [11], [13]. There were attempts to authenticate users on mobile devices using their keystroke dynamics with feature sets that are similar to those used in desktops [3], [10]. A more recent work, *KeySens* [4], has examined features unique to mobile devices. However, we leverage the proximity information to model the approach trajectory, whereas KeySens models the features only on the information which becomes available once the user's finger touches the screen.

Zhang et al. [17] developed a method for detecting when a user is using his left or right hand on a table-top multitouch surface based on human anatomy, work area, finger orientation and finger position. Unlike this work, their approach, which focused on bigger touch surfaces (table tops) and made a decision only on a single contact, did not use data from the finger approach trajectory.

In addition to the finger's approach trajectory, touch position, and tracking information, there is also information in the finger *posture*, which describes the finger's 3D orientation in relation to the touch sensor [16]. Posture information could

be complimentary to this paper's approach and could provide additional features to be used by our algorithm.

III. Data Collection

A. Software and hardware

We used a Samsung Galaxy S4 mobile phone in our experiments. This phone is equipped with a touch-screen sensor by Synaptics and runs Android OS. Standard Android APIs provide limited proximity information, only for UI elements which are specifically registered to receive such events. Our goal, however, is to monitor user touch patterns across various applications during normal phone operation. To achieve this, we wrote a data collection daemon which requires a modified kernel based on CyanogenMod [9]. The daemon records all events from the Linux input subsystem originating from the touchscreen sensor. The kernel modification is required to force the driver to send such events regardless of whether there are any active screen UI components requesting proximity information.

The driver reports touch information using a variant of the Linux Multi-touch (MT) Protocol, specifically, *Stateful Protocol "B"*. In addition to standard protocol events, an additional event *ABS_MT_ANGLE* with code *0x03c* is used to report touch ellipse orientation (yaw angle).

When one or more fingers are detected either touching the screen or in relatively close proximity, the information summarized in Table I can be obtained for each finger.

Parameter	Description
x,y	Finger position
z	Finger proximity (distance from the screen). 0 means touching.
a, b	Length of touch ellipse major and minor axis. Defined when $z = 0$
$angle$	Yaw angle of touch ellipse. Defined when $z = 0$

TABLE I: Touch parameters reported by driver

B. Data collection from human subjects

A total of 14 participants[1] (including the members of the research team) were presented with the modified Samsung Galaxy S4 and asked to follow a script containing different tasks. The total time needed to complete the entire script was approximately 15 minutes. The participants were asked to keep the phone in the *portrait* (vertical) mode.

The script included typical operations performed by a user on a regular basis. While the participant followed the instructions in the script and performed the tasks on the smartphone, the daemon running on the phone recorded the raw touch protocol events of the participant.

C. Touch data interpretation

Raw protocol data was decoded to create the sequence of touch measurements (Table I) constituting traces of the finger's motion before and after it touched the screen along with touch ellipse size and orientation.

[1]Institutional Review Board approval was obtained.

Finger approach trajectories look like those shown in Figure 1 and 2. The blue lines show the approach trajectory of the finger, and the green arrows show the projection of the finger velocity vector to the screen plane at the moment of the touch. The trajectories for two different users are visibly different. For example in Figure 1 the velocity vectors point to the left; whereas in Figure 2 the velocity vectors seem to point away from the center of the screen.

Empirically, we can estimate that a finger first registers when it is approximately 5mm from the screen. However, distance values reported by the sensor are in the 0 to 255 range, and it is unclear in what units the distance z from the screen is reported.

For estimating intruder detection time we calculated the average time duration between screen touches. The value is 0.406 touch events per second, or one touch in 2.46 seconds.

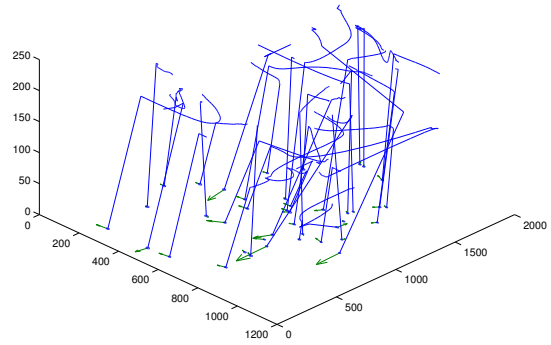

Fig. 1: Finger approach trajectories for user "A"

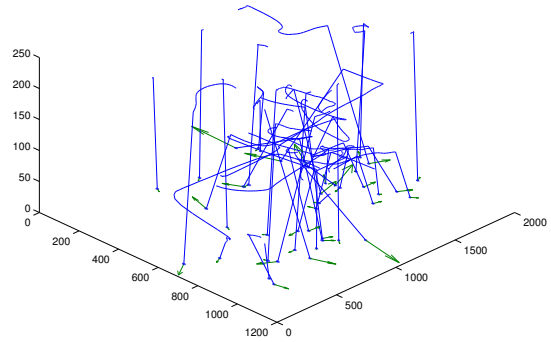

Fig. 2: Finger approach trajectories for user "B"

Since we wanted to analyze the trajectory of the finger approach, for each contact we extracted two measurements: one of the finger first touching the screen and the second one of the finger position just before that. By this, we limited the trajectory to just one measurement before the touch, essentially a vector, as shown in Figure 3. For illustration purposes, touch ellipse sizes were enlarged by a factor of 3.

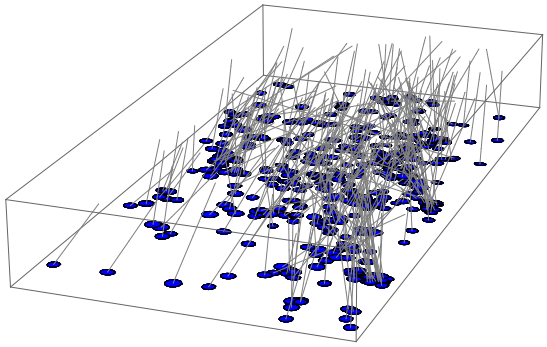

Fig. 3: Touch Vectors

This gives a set of *basic* features for each touch, summarized in Table II.

Features	Description
x, y	Touch coordinate
a, b	Touch ellipse axis length
px, py	Coordinates of projection of previous finger position before touch
pz	Distance to finger before touch
dt	Time before touch and previously registered proximity position
$angle$	Touch yaw angle in radians

TABLE II: Basic Features

An illustration of these values is shown in Figure 4.

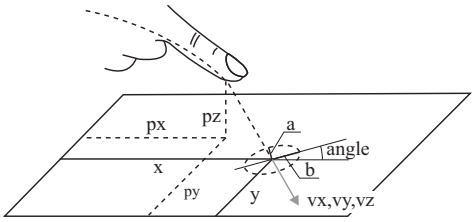

Fig. 4: Illustration of basic features

IV. Modeling User's Finger Approach

We developed several machine learning algorithms in order to model the user's "approach patterns." These can be trained online or offline. This work starts with offline training but could be extended to online training to automatically adjust the model for changing user touch patterns. The algorithms are discussed below.

A. Features

In addition to basic features which we extracted from protocol data, we have experimented with different representations and combinations of these features, for example conversion to spherical coordinates, numerical differentiation, and numerical integration. Table III summarizes all "synthetic" features we considered in addition to the basic ones.

To understand why these features should be different for different users, one can ponder their physical meanings. The magnitude of the velocity can differ if two people touch the screen very decisively or more slowly, while the direction of the velocity can show the direction of the finger; approach. For example, during data collection, we observed that some people held the device with their right hand and touched the screen with their left index finger, while others held the device with their right hand but touched the screen using their right thumb instead. Each of these two usage patterns can have different signatures for these values. In addition, the area of the touch ellipse can give some notion of the size of a user's finger.

Features	Description
dx, dy	Drift $px - x, py - y$
vx, vy, vz	Velocity $dx/dt, dy/dt, pz/dt$
θ_v, ϕ_v, r_v	Velocity (azimuthal angle, polar angle, radius)
θ_d, ϕ_d, r_d	Drift (azimuthal angle, polar angle, radius)
$area$	Finger contact area $(\pi * a * b)/4$
abx, aby, abz	Absement: $dt * dx/2, dt * dy/2, dt * pz/2$

TABLE III: Synthetic Features

While absolute coordinates (x, y, px, py) could provide some useful information, they are highly dependent on the actions the user performs and might not be generalizable especially when the training dataset is relatively small. In view of this, we decided to avoid using absolute coordinates.

B. Classification

Classifying known "owner" vs other possibly unknown users is known as an *anomaly detection* [8] problem. However we started with the simpler problem where we distinguish between the owner and other *known* users. That means that we have training data for all users and can apply supervised learning to build a *discriminative classifier*.

We will denote as $\vec{x}_1, ..., \vec{x}_n$ the set of feature vectors corresponding to touches made by the same user. We proceed under the *i.i.d* assumption, that $\vec{x}_i$ are mutually independent and governed by the same statistical distribution $\Pr(X)$. We will denote class $Y = \{+1, -1\}$ of each feature vector as $y_1, ..., y_n$. We will use the class label $+1$ for the owner and -1 for any other user. The classification will be performed by analyzing a single feature vector $\vec{x}_i$ and deciding whether it belongs to class $+1$ or -1. This is a standard two-class classification problem.

We tried two different approaches to classification: an Artificial Neural Network (ANN) and Support Vector Machines (SVM).

1) Artificial Neural Network: We used a feed-forward neural network with two hidden layers with 50-75 and 30 neurons respectively. The output layer used a *logistic sigmoid* activation function. We applied Principal Component Analysis (PCA) to the input set and discarded low-variance principal components, retaining 99.9% of the variance.

For ANN training, we split the data into 70% training and 30% test sets. Since we had to assign only one user to class $+1$ and all remaining users to class -1, the ratio of data points belonging to each class was approximately 4:13. In such a situation, it is insufficient to look at a simple metric like *miss rate* to evaluate classifier performance, so we also

used *precision*, *recall*, and *F1 score* metrics. The resulting performance of the ANN classifier is shown in Table IV.

Feature Set	Miss Rate	Precision	Recall	F1
$angle$, $area$, nx, ny, nz	0.007	0.987	0.992	0.989
$angle$, $area$, abx, aby, abz	0.007	0.989	0.991	0.990
$angle$, $area$, r_v, θ_v, ϕ_v	0.001	0.998	0.999	0.999

TABLE IV: ANN Results for different feature sets

2) Support Vector Machines: In addition to the ANN classifier, we also tried to implement an SVM classifier. We tried various parameters and kernels including *linear*, *polynomial*, and *radial basis* on different feature sets. Unfortunately, we were not able to achieve good classification accuracy. The best *F1-score* we saw was 0.67 (compared to 0.99965 by ANN on the same data).

C. Sequential classification

In Section IV-B, we attempted to classify the user using a single feature vector. However, we are not bound to classification of users based on a single touch. To improve accuracy, we can treat x as a sequence of events from which we can analyze several touches before making the decision. We would like to classify using a minimal number of samples which can provide enough information to make a decision within predefined confidence bounds. The smaller the number of samples required, the faster the system is able to detect an unauthorized user.

The Sequential Probability Ratio Test (SPRT) was introduced by Wald [15]. There were attempts to use it for sequential classification, for example [14]. We will assume that the given sequence of observations $\vec{x}_1, ..., \vec{x}_n$ belongs to an unknown class y. Our *null hypothesis* $\mathscr{H}_0$ would be that $y = +1$, and our *alternative hypothesis* $\mathscr{H}_1$ would be that $y = -1$. SPRT provides an optimal sequential decision strategy S_n defined as:

$$S_n = \begin{cases} \textit{accept } \mathscr{H}_0 & R_n \leq B \\ \textit{accept } \mathscr{H}_1 & R_n \geq A \\ \textit{undecided} & B < R_n < A \end{cases} \tag{1}$$

$$R_n = \frac{\Pr(x_1, ..., x_n \mid y = -1)}{\Pr(x_1, ..., x_n \mid y = +1)} \tag{2}$$

Constants A and B are chosen based on required rates of *errors of the first and second kinds* α and β respectively. In practice, A and B are difficult to calculate precisely, and the following approximation for their values is suggested by Wald:

$$A' = \frac{1-\beta}{\alpha} \quad B' = \frac{\beta}{1-\alpha}$$

Wald shows that A' and B' defined this way provide upper and lower bounds for R_n for given α and β. Moreover, he shows that when A' and B' are used, the actual error rates α' and β' relate to the original α and β in the following way:

$$\alpha' + \beta' \leq \alpha + \beta$$

The SPRT strategy is applied by increasing the number of observations n until a decision is reached.

With the i.i.d assumption, we can rewrite Equation (2) as:

$$R_n = \frac{\prod_{i=1}^{n} \Pr(x_i \mid y = -1)}{\prod_{j=1}^{n} \Pr(x_j \mid y = +1)}$$

Applying the Bayes theorem in both numerator and denominator and simplifying we get:

$$R_n = \frac{\prod_{i=1}^{n} \Pr(y = -1 \mid x_i)}{\prod_{j=1}^{n} \Pr(y = +1 \mid x_j)} \left(\frac{\Pr(y=+1)}{\Pr(y=-1)}\right)^n \tag{3}$$

It has been shown [2] that outputs of a simple 2-class feed-forward ANN classifier can be interpreted as class probabilities. Therefore, for each x_i, the ANN classifier provides us with $\Pr(y = +1 \mid x_i)$ and $\Pr(y = +1 \mid x_i)$. That means, we can use these outputs in the first term of Equation (3). Since we have two mutually exclusive classes, Equation (3) could be further rewritten as:

$$R_n = \prod_{i=1}^{n} \left(\frac{1}{\Pr(y = +1 \mid x_i)} - 1\right) \left(\frac{1}{\Pr(y=-1)} - 1\right)^n \tag{4}$$

For iterative computation, it is more convenient to work with logarithms of R_n, A, and B. Our decision strategy from Equations (1) and (3) could be written as:

$$S_n = \begin{cases} \textit{accept } \mathscr{H}_0 & \log R_n \leq \log B \\ \textit{accept } \mathscr{H}_1 & \log R_n \geq \log A \\ \textit{undecided} & \log B < \log R_n < \log A \end{cases}$$

$$\begin{aligned} \log R_n = &\sum_{i=1}^{n} \log \Pr(y = -1 \mid x_i) \\ &- \sum_{i=1}^{n} \log \Pr(y = +1 \mid x_i) \\ &+ n \log \Pr(y = +1) - n \log \Pr(y = -1) \end{aligned}$$

The only remaining unknowns are class priors $\Pr(y = -1)$ and $\Pr(y = +1)$. It is difficult to estimate these from the training data. However, we can use some prior knowledge to estimate them for use as our model parameters. For example, in our scenario of classifying the user as *owner* vs. *intruder*, these priors describe our risk assessment or our belief of how likely it is that the phone has been stolen. They could be set based on the environment. For example, if we know the phone location (using GPS or WiFi fingerprinting), we can argue that while it is at the user's home, it is less likely that the phone has been stolen than if the phone is at some other location. We can also use other factors, such as how recently the phone was unlocked using the user's secret password and last numbers called. The phone is less likely stolen if the last number called comes from the address book or has been frequently called previously. In the worst case, if we know nothing about the environment, the priors could be equally probable, and both may be set to 0.5.

The selection of priors has an effect on decision time, which some people might consider counter-intuitive. Let us denote the ratio or priors from Equation (3) as:

$$R_p = \frac{\Pr(y = +1)}{\Pr(y = -1)}$$

Based on the value of R_p, we can distinguish three types of priors. We will call them *uniform* when $R_p = 1$, *safe* when $R_p > 1$, and *risky* otherwise (when $R_p < 1$).

If we simulate *owner* and *intruder* detection for all 3 types of priors, the results will look something like that shown in Figure 5. For simplicity, we have assumed that classifier outputs $\Pr(y = +1 \mid x_i)$ and $\Pr(y = -1 \mid x_i)$ are constat for all x_i As we can see from the plot, *risky* priors speed

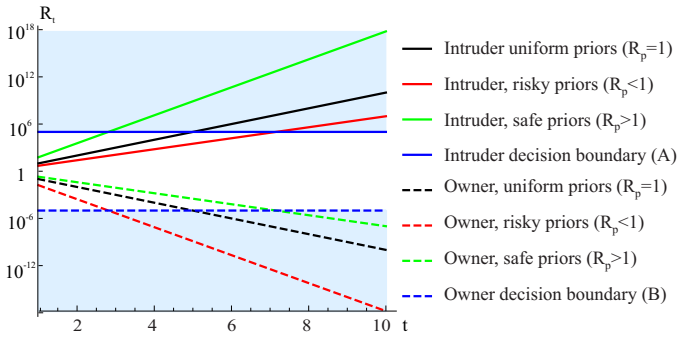

Fig. 5: Effect of priors on decision time

up owner identification but slow down intruder detection. On the other hand, *safe* priors speed up intruder detection but delay owner identification. A common-sense explanation for this effect is that the model, expecting a certain type of user, is faster in reacting to evidence to the contrary. For example, seeing suspicious behaviour which is not expected in a safe environment triggers the alarm faster.

One way to look at this is by rewriting Equation 3 as follows:

$$R_n = \frac{\prod_{i=1}^{n} \frac{\Pr(y=-1|x_i)}{\Pr(y=-1)}}{\prod_{j=1}^{n} \frac{\Pr(y=+1|x_j)}{\Pr(y=+1)}}$$

This clearly shows that we normalize conditional probabilities of each individual sample by dividing them by their respective class priors.

The effect of priors and the desired error rate could be illustrated by a join probability density function of decision time (proportional to n) and *error rate* shown in Figures 6 and 7 sampled from the experiments we performed. For simplicity here, we set both SPRT parameters α and β to the same value error_rate.

D. Dominant hand detection

An interesting side result of our analysis of proximity information is that we can easily distinguish between left and right handed users using their dominant hand. This can be done based on a sign of the dx feature. As a left handed user's finger usually approaches from the left side before touching the screen, such users' dx will have a negative sign. This correlation could be seen in Figure 8 which depicts the PDF of dx values for left handed users (shown with dashed red lines) and for right handed users (shown with solid black lines).

The sign of the sample mean of dx values from multiple samples provides a very good indicator of whether the user is

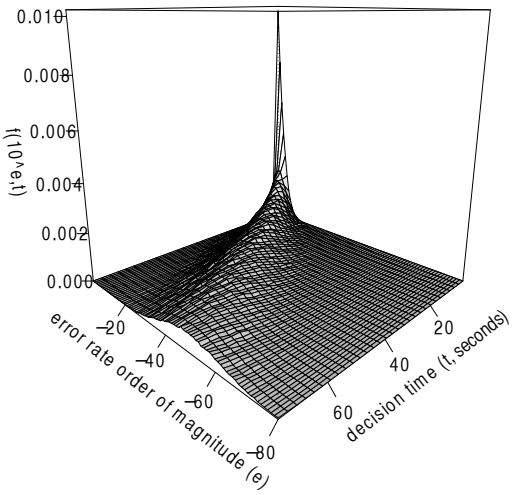

Fig. 6: $y = -1, \Pr(y = -1) = 0.9$. PDF showing the effect of desired error rate and decision time.

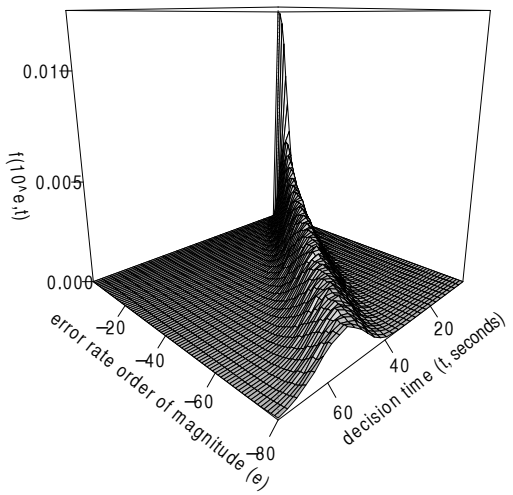

Fig. 7: $y = -1, \Pr(y = -1) = 0.1$. PDF showing the effect of desired error rate and decision time.

left or right handed. For all our subjects we can see a 100% correlation between the $\bar{dx}$ sign and the dominant hand.

Since dominant hand detection was not the primary topic of our study, the results presented above are preliminary. Additional research is required to provide more rigorous methodology.

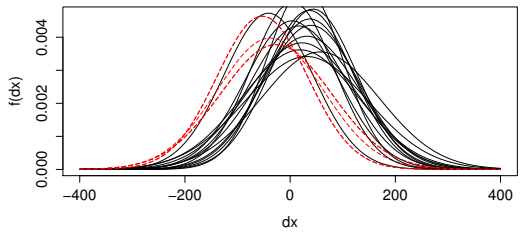

Fig. 8: PDF of finger drift

V. Experimental Results

Figure 9 shows a simulated scenario of user change. The dataset was composed from three segments of equal lengths. The data in the first and the third segments belongs to the user with class $y = -1$, and the data in the second belongs to the user with class $y = +1$. The purpose of this experiment was to investigate how fast our algorithm detects when the phone is transferred to a different user (simulated phone loss followed by recovery).

Green circles show probabilities as calculated by ANN for 10% of the individual samples. The black line shows the classification according to our algorithm. In this experiment, the following parameters were used[2]: $\Pr(y = +1) = 0.5, \Pr(y = +1) = 0.5, \alpha = 10^{-20}$, and $\beta = 10^{-20}$. The purple, dotted line shows the probability of the class $+1$. As we can see, the algorithm performed very well and correctly identified users.

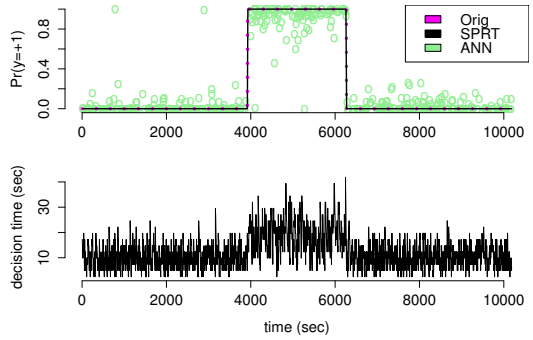

Fig. 9: User change detection

The time required to make the classification decision is shown in Figure 9. As we can see, on average, it takes slightly less time to make decisions for class -1 and more for class $+1$. This could be attributed to the disproportionate amount of data per class used to train the ANN classifier.

Overall system performance in making a decision is shown in Figure 10. It shows a histogram of the number of touches a user needs to perform before he or she is classified as an *owner* or an *intruder*. This was calculated for users across both classes. On average, a decision takes 5 touches or 12.6 seconds. The maximum duration was 20 touches or 49 seconds.

[2]These parameters exceed requirements of The European standard for access-control systems EN-50133-1 which mandates a false-alarm rate of less than 1%, with a miss rate of no more than 0.001% [8]

We find this decision time satisfactory for practical purposes, as an intruder could be detected in less than a minute which is insufficient time do much damage to the stolen phone.

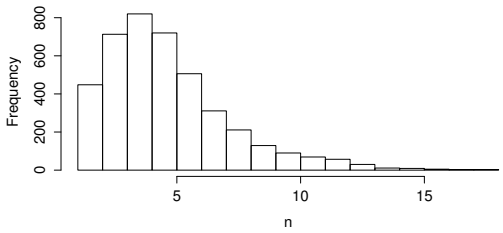

Fig. 10: SPRT classifier decision time

References

[1] Banerjee, S. P., and Woodard, D. L. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research 7*, 1 (2012), 116–139.

[2] Bishop, C. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[3] Clarke, N. L., and Furnell, S. M. Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security* (2007).

[4] Draffin, B., Zhu, J., and Zhang, J. KeySens : Passive User Authentication through Micro-behavior Modeling of Soft Keyboard Interaction. In *Proceedings of MobiCASE 2013, Fifth International Conference on Mobile Computing, Applications and Services* (2013).

[5] Feng, T., Yang, J., Yan, Z., Tapia, E., and Shi, W. TIPS: Context-Aware Implicit User Identification using Touch Screen in Uncontrolled Environments. In *HotMobile* (2014).

[6] Frank, M., Biedert, R., Ma, E., Martinovic, I., and Song, D. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *Information Forensics and Security, IEEE Transactions on 8*, 1 (Jan 2013), 136–148.

[7] Giot, R., El-Abed, M., and Rosenberger, C. Keystroke dynamics with low constraints svm based passphrase enrollment. In *Biometrics: Theory, Applications, and Systems, 2009. BTAS '09. IEEE 3rd International Conference on* (2009), 1–6.

[8] Killourhy, K. S., and Maxion, R. a. Comparing anomaly-detection algorithms for keystroke dynamics. *2009 IEEE/IFIP International Conference on Dependable Systems & Networks* (June 2009), 125–134.

[9] Kondik, S., and community. CyanogenMod. android community ROM based on Jelly Bean", 2013. http://www.cyanogenmod.org/.

[10] Maiorana, E., Campisi, P., Gonzlez-Carballo, N., and Neri, A. Keystroke dynamics authentication for mobile phones. *SAC 2011* (2011).

[11] Maxion, R. A., and Killourhy, K. S. Keystroke biometrics with number-pad input. *2010 IEEE/IFIP Int. Conf. Dependable Syst. Networks* (June 2010), 201–210.

[12] Mock, P., Edelmann, J., Schilling, A., and Rosenstiel, W. User identification using raw sensor data from typing on interactive displays. *Proc. 19th Int. Conf. Intell. User Interfaces - IUI '14* (2014), 67–72.

[13] Peacock, A., Ke, X., and Wilkerson, M. Typing patterns: A key to user identification. *IEEE Security Privacy* (2004), 40–47.

[14] Sochman, J., and Matas, J. WaldBoost Learning for Time Constrained Sequential Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) 2*, 150–156.

[15] Wald, A. *Sequential analysis*. Courier Dover Publications, 2004.

[16] Zaliva, V. 3D finger posture detection and gesture recognition on touch surfaces. In *Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on*, IEEE (2012), 359–364.

[17] Zhang, Z., Zhang, F., Chen, H., Liu, J., Wang, H., and Dai, G. Left and right hand distinction for multi-touch tabletop interactions. *Proc. 19th Int. Conf. Intell. User Interfaces - IUI '14* (2014), 47–56.