TapDynamics: Strengthening User Authentication on Mobile Phones with Keystroke Dynamics

Grant Ho

1 INTRODUCTION

The convenience and ubiquity of mobile devices make them a rich source of personal data such as email and financial information. Unfortunately, this data is often guaded by only a simple drawing pattern or short PIN code, which recent literature has shown to be woefully insecure [1]; as a particularly scary example, Bonneau's recent work suggests that an attacker can guess most users' PIN codes after only eleven attempts [2]. With the the rise of smartphone theft, we see a crucial need for stronger security mechanisms that protect a user's data on smartphones. Our project aims to address this problem by strengthening user authentication when a person unlocks/logs into a phone. Specifically, we construct and analyze four keystroke dynamic classifiers that use a smartphone's sensors to learn the key tap behavior of the true owner. After this training, our classifiers can be used to determine whether a login attempt is being made by the true owner or by an attacker who has guessed the owner's PIN.

2 BACKGROUND

The study of leveraging a user's typing behavior to strengthen passwords is known as "keystroke dynamics"; over one hundred papers have been published on keystroke dynamics [3]. Much of this work has focused on desktop keyboards and examines three features: the duration of each key press, the latency between keystrokes, and implicit measures of keystroke force through things like computer microphones [4] [5]. More recent work has examined deploying keystroke dynamics on mobile devices [6] [7]; however, this work continues to use only keystroke timing features and often focuses on passwords that are ten characters or longer. Our work is different from the existing literature because we are the first to examine how to capitalize on new sensor data on modern smartphones in order to apply keystroke dynamics to short, four-digit PIN codes.

3 METHODOLOGY

3.1 Threat and Defense Models

Our project addresses the following threat model:

1. Alice has a mobile phone that is secured using a standard lock screen with a four digit PIN
2. An attacker (a thief) manages to steal Alice's phone through pickpocketting or common snatch-and-run theft on a bus or train.
3. The attacker now attempts to break into Alice's phone by iterating through common PINs or guessing her PIN based on profiling and searchable online information.

As stated earlier, our defense is a keystroke dynamics classifier that learns to distinguish between Alice attempting to unlock her phone and someone else attempting to unlock her phone.

3.2 Data Collection

We built an open source, Android login application [8] that collects the following biometric features when a user enters in a PIN code: the duration of each key tap, the latency between each key tap, the size of each key tap, and all accelerometer readings over the course of a login attempt. Our Android app randomly assigns a user one of five PIN codes to enter for thirty times; these thirty "trials" were separated into two blocks of fifteen with a thirty second pause in between each set of fifteen trials. We randomly assigned each user a PIN code from a fixed list of five PIN codes; three of these PIN codes are the most popular PIN codes and account for 18% of all user's PIN codes [1] and the remaining two PIN codes were randomly generated. Unfortunately, crowd sourcing services like Mechanical Turk prohibit tasks from requiring users to download/install any software (such as Android apps), so we were required to manually gather data for this project; our procedure was submitted to Stanford's IRB and received clearance as non-human subject research because we randomly assign each user an ID number and do not collect any personally identifiable information. Since receiving IRB approval in November, we have gathered data from approximately fifty five unique users which resulted in 1704 data samples.

3.3 Preprocessing

Each training example (login attempt) consists of five key taps: one tap for each digit in the PIN and a final tap for the 'Enter' key. This resulted in five features for duration, four features for latency, and five features for size. For the accelerometer readings, we found that our data had a variable number of readings per tap and across training examples, making the raw accelerometer data infeasible for direct use as a feature. We originally looked at incorporating accelerometer readings as a feature because we theorized that a user might have a natural orientation for holding a phone during use; this natural orientation would then be perturbed by the force of each key tap. Our hope was that the accelerometer readings might be able to construct an orientation and force profile that would be unique to each user. This project makes a first attempt at generating this accelerometer profile by computing various statistics over all accelerometer readings in a login attempt to create a total of twenty-one accelerometer features per training example; specifically, we compute statistics like the mean, min, max, variance, first quartile, second quartile, and third quartile for the x,y, and z components over all accelerometer readings in a training example. Thus, each data sample consist of thirty-five features, which we extracted from the raw sensor data that our test phone collected.

3.4 Classification Techniques

We tested four classifiers for our experiments. Our first classifier was the Manhattan Distance Classifier, which is a statistical classifier that computes the distance between the mean feature vector of a user's training data and the feature vector for an attempted login; if the distance between the login attempt and the existing user's data is within a certain statistical threshold, the classifier allows the attempt to successfully login. Our second classifier was the Random Forest Classifier, which generates an ensemble of decision trees that vote on a decision based on the feature data in our training sample. We chose these first two classifiers because they are highly regarded in the literature [9] [10]. Additionally, we tested Gaussian Discriminant Analysis because many of our features appear to be normally distributed. Finally, we also tested an SVM with a linear kernel after discussions with the course staff and determining that the literature sparsely mentions and shallowly assesses the efficacy of SVMs.

4 EVALUATION

For this project, we downloaded the data off of our phones and used a Ruby script to process this raw SQLite data into a CSV file for analysis in MATLAB. To evaluate our construction, we use two commonly used metrics in keystroke dynamics that assess the security and usability of our models.

1. False Acceptance Rate (FAR) = the percent of attacker data samples that we incorrectly label as the true user (percent of attackers we accidentally accept). FAR allows us to measure the security of our construction, since a low FAR means we are good at rejecting attackers.

2. False Rejection Rate (FRR) = the percent of true user data samples that we incorrectly label as an attacker sample (percent of users we accidentally lock out). Thus, FRR measures the usability of our construction.

We used the following procedure to generate our results:

1. For each PIN code p, we selected all data for users with PIN = p. Call this dataset 'D'.

   (a) For each user in 'D', the first 15 samples became the positive training set and the remaining 15 samples became the positive test set. We chose to split the dataset in this manner because during data collection, our app inserted a 30 second pause between the first 15 and second 15 login trials. We wanted to use this pause to test if we could still distinguish the true user after time has passed between login attempts, which models real world where Alice will make one login attempt, use her phone, and then make the next login attempt later on when she needs her phone again.

   (b) To generate our negative data sets, the first 15 samples from all other users in 'D' became our negative training set while the remaining 15 samples for all other users in 'D' became our negative test set.

   (c) Next, for each user in 'D', we trained and tested each classifier and computed the average FAR and FRR among all users in 'D' to get the FAR and FRR for each PIN.

2. Finally, since each PIN in our dataset has ten or eleven users each, we simply averaged the FAR and FRR over the five PINs to obtain the FAR and FRR for each classifier.

The following chart presents the overall FAR and FRR for our four classifiers (which closely mirror the FAR and FRR for each PIN).[1]. Because of these results and paper length constraints, the rest of this paper focuses on GDA and SVM.

Model	FAR	FRR
Manhattan Distance	10.1%	21%
Random Forest	1.4%	33.0%
GDA	1.9%	20.8%
SVM (Linear Kernel)	0.9%	20.9%

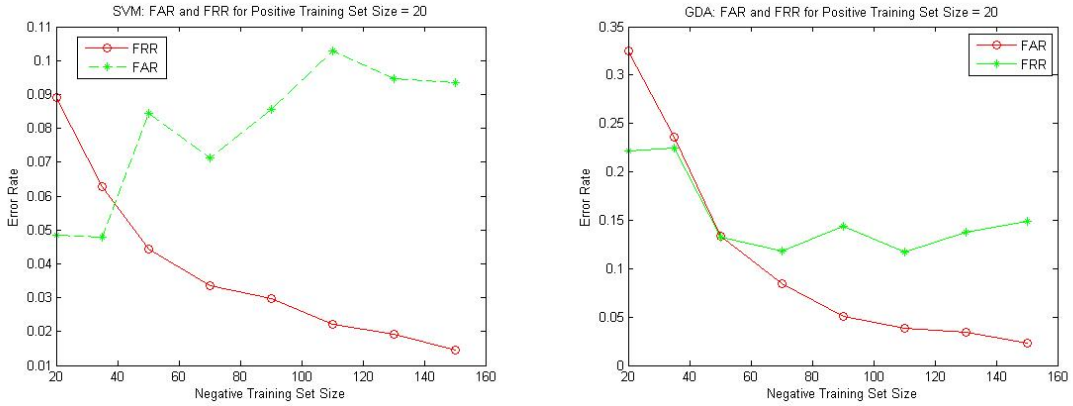
Table 1: Model Error Rates (FAR = false acceptance rate and FRR = false rejection rate)

These results suggest that our top classifiers offer excellent security in terms of detecting and rejecting attackers. Unfortunately, the relatively high FRR's raise concerns about the usability of our keystroke dynamics construction because people might not want to use a system that accidentally locks them out of their phone 20% of the time. Thus, we looked for ways to improve the high FRR's.

First, we realized that our data set is highly skewed with many more negative training examples than positive examples. For each user, we divided the person's 30 trials into a training and test set of 15 positive samples; however, our negative sets are generated by taking 15 samples from all other users with the same PIN, resulting in 15 * 10 = 150 negative training examples. We theorized that this overwhelming number of negative training examples might explain our high FRR because it inflates the prior attacker class probability

[1]The Manhattan Distance classifier is incredibly time intensive to run, so we only obtained FAR's and FRR's for the pins 1111 and 1212; however, we feel comfortable reporting results in terms of total averages because even on these two pins, Manhattan Distance was clearly outperformed by the others (e.g. Manhattan on "1111" has FAR = 9.1% vs. GDA FAR = 2.6% and FRR = 24% vs. GDA FRR = 18%)

[p(y = attacker)] and biases our classifier to reject a login attempt. To test this, we varied the negative training set size by randomly selecting a certain number of login attempts from our pool of negative/attacker samples and compared the FAR/FRR for our classifiers against these different negative training set sizes (15, 35, 55, 75, 95, 105, 125, 150). These results showed that balancing our positive and negative training set sizes yield significantly better false rejection rates (e.g. for our SVM, it lowered our FFR from 20.9% at 150 negative examples to 13.1% at 15 negative examples). Unfortunately, this came at the expense of increasing our false acceptance rate (0.9% to 11.9% for our SVM).[2] To gain more insight, we compared our test errors (both FAR and FRR) with our trianing errors; for each training set size, we found that our training error was approximately zero for both FAR and FRR - dramatically better than our test error. This suggested that our models were suffering from high variance. Thus, to counteract this extreme overfitting, we increased our positive training set to include the first twenty samples from each user (leaving our test set with ten samples per user) and obtained the following plots:



From these results, we see that increasing the number of positive training examples helps correct overfitting. This technique appears to be most beneficial to SVMs because it reduced the FRR while maintaining the FAR within one or two percentages. These two optimzations achieved satisfying results for our SVM; using 20 positive training examples and 40 negative training examples, our SVM has a FAR under 6% and a FRR under 5%. We consider these to be extremely good results given that we're using incredibly short PINs and our results beat the FAR/FRR's achieved in most existing literature (10-20%), even though the literature often uses PINs or passwords of at least ten characters long [6] [7]. In light of these satisfying results, we conducted an ablative feature analysis on our SVM to determine the importance of each feature. Based on the optimal balance between FAR and FRR from our plots, we performed our ablative analysis on a training set of 20 positive examples and 40 negative training examples.[3]

From the following table, we see that our tap size and accelerometer features boost our SVM's false acceptance rate from 17.8% to 4.4% and improve its false rejection rate from 14.7% to 5.3%. Thus, we conclude that our hypothesis was correct: new sensor data on smartphones provide features that can increase the security and usability of keystroke dynamics on smartphones.

Features Removed (Cumulative)	SVM FAR	SVM FRR
Baseline (no features removed)	4.4%	5.3%
Accelerometer Statistics	11.7%	12.6%
Key Tap Sizes	17.8%	14.7%
Key Tap Duration	28.4%	17.4%

[2]We withhold tables and plots for these in-between analyses because of page limit restrictions

[3]We tried techniques to improve the performance of GDA, like outlier removal to correct overfitting to outliers and make our feature data more Gaussian; but, these techniques didn't yield significant improvement and SVMs still outperformed it.

5 CONCLUSION

Our SVM's excellent results show that keystroke dynamics can be an effective means of enhancing the security of a user's data on smartphones; even on an extreme PIN of "1111", our SVM achieves remarkable results with a 5.6% FAR and a 7.6% FRR. With an overall false acceptance rate of 4.4%, password guessing becomes an difficult way for thieves to break into your phone and access your data; even if an attacker correctly guesses your PIN, our classifier will likely reject the attacker based on his anomalous tap dynamics. Moreover, our false rejection rate of 5.3% seems to be low enough for this system to usable on real-world smartphones. Finally, our ablative feature analysis shows that features like tap size and accelerometer readings improve the efficacy of keystroke dynamics by over 50%, which validates our hypothesis that smartphones sensors offer rich, new features that have been underexplored in the literature.

For future work, our analysis indicates that increasing the number of positive training examples (the number of true owner login examples) improves the usability of our classifier. Thus, future work should explore techniques like SMOTE to create synthetic positive training examples that help balance the training sets. Additionally, we intend to explore different ways to extract features from our accelerometer data; right now we simply use rough statistics to extract accelerometer features. While this significantly contributes to our SVM, we found that it actually caused worse performance (overfitting) on other models like GDA (not shown in this paper because of length constraints). Using fewer, but more intelligent features from the accelerometer data might improve the utility of this feature. Similarly, an interesting machine learning question to explore would be why our GDA model performs worse when we balance the positive and negative training set sizes; we theorized that balancing the training set sizes would at least improve the FRR because its prior class probabilites would be more balanced, but that proved to be false.

6 ACKNOWLEDGEMENTS

I would like to thank Dieterich Lawson for writing the scripts for our Manhattan Distance and Random Forrest classifiers and the Ruby script to convert our sqlite database into a CSV of features. I would also like to thank Sameep Tandon for many insightful discussions on different machine learning techniques to try as well as Dan Boneh for the original idea and recurring advice on this project.

References

[1] DataGenetics. (2012, Sep.) Pin analysis. [Online]. Available: http://www.datagenetics.com/blog/september32012/

[2] J. Bonneau, S. Preibusch, and R. Anderson, "A birthday present every eleven wallets? the security of customer-chosen banking pins," in *Financial Cryptography and Data Security*. Springer, 2012, pp. 25–40.

[3] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*. IEEE, 2009, pp. 125–134.

[4] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication," *Future Generation Computer Systems*, vol. 16, no. 4, pp. 351–359, 2000.

[5] J. Roth, X. Liu, A. Ross, and D. Metaxas, "Biometric authentication via keystroke sound."

[6] A. Buchoux and N. L. Clarke, "Deployment of keystroke analysis on a smartphone," in *Australian Information Security Management Conference*, 2008, p. 48.

[7] E. Maiorana, P. Campisi, N. González-Carballo, and A. Neri, "Keystroke dynamics authentication for mobile phones," in *Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, 2011, pp. 21–26.

[8] G. Ho. [Online]. Available: https://github.com/grantho/TapDynamics/

[9] P. S. Teh, A. B. J. Teoh, and S. Yue, "A survey of keystroke dynamics biometrics," *The Scientific World Journal*, vol. 2013, 2013.

[10] R. A. Maxion and K. S. Killourhy, "Keystroke biometrics with number-pad input," in *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*. IEEE, 2010, pp. 201–210.