Continuous Transparent Authentication with User-Device Physical Unclonable Functions (UD-PUFs) based on Mobile Device Touchscreen Interactions

Timothy M. Dee
Electrical & Computer Engineering
Iowa State University
Ames, IA, USA
deetimothy33@gmail.com

Ian T. Richardson
Electrical & Computer Engineering
Iowa State University
Ames, IA, USA
ian.t.rich@gmail.com

Akhilesh Tyagi
Electrical & Computer Engineering
Iowa State University
Ames, IA, USA
tyagi@iastate.edu

ABSTRACT

In this paper we discuss the problems with existing authentication schemes. We then implement a system which seeks to solve problems present in these other systems. This system uses an $n-gram$ Markov model to track properties of a user's interactions with the soft keyboard of a mobile device. This system is used to continuously compare a user's current behavior against the past behavior of that same user. We describe how our system protects against the vulnerabilities existing in current authentication systems and demonstrate the practicality of our implementation.

1. INTRODUCTION

Malicious parties attempt to gain access to data of their victims. Attackers go through the trouble of preforming these attacks because gaining access to the data might be quite lucrative. Mobile devices are often used to store credit card information, passwords, and banking information. This data can be leveraged for significant loss.

Many current practices keep mobile devices secure with a lock screen. These protection mechanisms require that the user preform some action when the mobile device state moves from inactive to active. This one-time authentication allows access to everything on the device including all applications. Significant amounts of data is hosted in various applications on mobile devices. This data could be potentially damaging to the user if it is exposed to a malicious party. The compromised information could include social security numbers, credit cards, passwords, and banking information.

The tendency for applications to remember a user's authentication information functions to make the lock screen the only line of defense in many cases. The combination of having a single authentication with a significant amount of valuable data makes mobile devices potentially very lucrative to steal. The mobile nature of these devices exacerbates the issue causes them to be extremely susceptible to theft. In fact, mobile device theft is a serious problem. Consumer Reports [1] reported over 3.1 million smart phone thefts in 2013. Federal Communications Commission (FCC)[2] in its December 2014 report estimates 368.9 phone thefts per 100000 individuals in 2013. It states that about one third of crime involves a mobile phone. There are many ways an attacker might exploit this one-time authentication in order to gain access to resources on a mobile device. There are several different types of lock screens which are commonly employed. Android Version 6.0.1 Marshmallow includes password, pin, and pattern options for securing the lock screen. Pin allows the user to create a small fixed-length sequence of numbers. Password authentication is similar to pin authentication except letters, symbols, and numbers are all used to create the secret. Swipe authentication is seemingly different allowing the user to create a pattern of swiping between 9 points on the screen. These methods are based on the user having some secret knowledge which is entered on the lock screen using the touchscreen. This authentication occurs each time the phone state changes from inactive to active.

All methods of authentication mentioned discussed thus far require that a user enter information using the touch screen each time they want to utilize their device. There are some problems with this. Malicious parties may try to exploit the requirement of entry through the touch screen by shoulder-surfing. This practice involves an attacker watching a person enter their secret so that it might be replicated later. [3] discusses the vulnerability of various touch screen input methods to shoulder-surfing attacks while [4] measured the susceptibility of users to shoulder surfing attacks observing attackers having a ($> 20\%$) success rate on Android keyboards in the reproduction of a password after 3 attempts. Smudge attacks where, oily residues are used to detect common touch screen patters, also represent a significant security risk for secrets input via the touch screen. [5] explores the feasibility of such attacks finding in one experiment the lock screen pattern was partially identifiable 92% of the time and full identifiable 68% of the time.

These vulnerabilities have led to the development of lock screens which use a user biometric in an attempt to increase security. Facial recognition and finger print scanners have been employed as lock screen mechanisms. These methods

insufficient to protect the device from attack. [6] demonstrates a method to exploit the finger print scanner on an android device using printed fingerprints. This method is simple and fast as long as a print of the authentic user's finger can be obtained. Other works by Germany's Chaos Computer Club [7] have show that it is possible to lift a finger print belonging to the authentic user from the device touch screen and use it to pass the finger print scanner authentication on Apple's Iphone $4S$ and 5. Attacks against facial recognition schemes involve spoofing the facial features of the authentic user in order to pass facial recognition authentication. [8] accesses the practically anti-spoofing measures in real word scenarios observing low generalization of and possible database bias in existing schemes.

In many situations, an attacker may not even need to pass the lock screen authentication. Imagine a situation where the authentic user of a device preforms the authentication, unlocking the device. The device then falls into the hands of an attacker. This attacker can not only exploit the authenticated state of the device, but may use the authenticated state to disable the lock screen entirely; Thus allowing permanent access to all applications on the device. [9] surveyed the lock screen behavior of 320 individuals ages $18-67$ with a median age of 31. Of these individuals, 39.6% rated themselves as having a very high level of IT expertise. The results of this survey indicate 56.3% of the participants did not use a lock screen at all.

This paper does not propose a new method of authenticating the user through the lock screen. Instead we seek to mitigate risk of loss should a device become compromised. We describe a continuous authentication layer which can distinguish among users using a biometric of the user generated though data entry on an Android device's soft keyboard. In many situations there exists a trade off between security and convenience from the user's perspective. This trade off can be observed in the Android lock screen. The necessity to enter a pin, password, or pattern every time access to the device is desired is an inconvenience. In exchange for this inconvenience, users are rewarded with some security. The solution presented in this paper provides additional security while requiring no additional actions from the user. This is accomplished by recording a biometric of the user's interactions with the touch screen in the background. The user continues, unaffected by this collection, interacting with the device as they normally would. The most resent interactions are then tested against previous actions to determine if the user's pattern has begun to deviate. Deviations may indicate that a malicious party has gain control over the device.

We provide a step forward, in mobile device security by establishing a continuous, unobtrusive user identity. This identity enhances the ability of mobile devices to recognize when the device may have come under the physical possession of an illegitimate user. This system provides innovations in the following areas:

- We show that touch screen interactions may be used in order to distinguish a legitimate user of a mobile device apart from illegitimate users. Specifically, section 3 shows how a combination of a touch screen pressure and the location on the screen may be used to develop a model of that user's behavior.
- We establish the behavior of a user on a device is unique to that user. Many applications, such as Google Wallet, can benefit from an increased assurance the user is authentic. Section 4 establishes that a difference in user identity may be detected by our implementation.
- We establish the behavior of a user on a device is unique to that device. Such an identity could be useful in defeating attacks where a user is tricked into entering secret data, such as a password, on another device. Section 4 establishes that a difference in device identity may be detected by our implementation.
- The system is unobtrusive meaning there is no convenience cost assessed upon the user. All security improvements are accomplished without requiring any additional actions from the user. Our implementation, capable of utilizing touch screen data collected in the background, is described in Section 3
- The performance of this system has sufficiently high accuracy, $(70-90+\%)$, and low computational overhead, $(1-5sec)$, to make it practical. A performance comparison is presented in Section 4 which demonstrates the computation time as a function of the number of touch screen interactions used and the running time of the system on a Nexus 7 tablet.
- This work solves the problem of detecting a non-authentic user in cases where the lock screen has been bypassed. Other solutions fail to consider this problem or fail to provide a sufficient solution as discussed in Section 2.

In deciding if a user is legitimate, it is useful to define three categories: something the user knows, something the user has, and something the user is. Traditional mobile authentication schemes, such as a lock screen, utilize only something a user knows. Interactions between a user and the touchscreen of a mobile device are rich with information. Current solutions suffer from underutilization of this information; they discard much of the content of these interactions in favor using the location of the interactions exclusively. Our system utilizes pressure, time, and location capturing the currently under-utilized potential of these interactions to expose patterns unique to a user.

We use these properties to construct a model of how the user interacts with a mobile device. This model takes as input a sequence of touches preformed by the user. From this sequence probabilities are developed which represent the likelihood of a given touch screen interaction based on the properties of a number of previous interactions. We use an $n-gram$ Markov model, a subclass of Markov text analysis. This model uses n previous states in computing next state probabilities. This model is explained more throughly in Section 3.

2. THE PROBLEM

Current protection mechanisms do not adequately protect data contained on mobile devices. The current scheme provides a single line of defense, the lock screen. If this line of

defense has been compromised by an attacker, all applications and data on the device become available. This scheme might be considered successful if the lock screen offered a high level of device security. Unfortunately, there are exploits for many common lock screen authentication mechanisms allowing an attacker to bypass the protection. Common protection mechanisms include pin, password, pattern, finger print scanners, and facial recognition. The first three can be overcome successfully with shoulder-surfing [3] and smudge attacks [5]. Finger print scanners and facial recognition require a different approach where the relevant features of the legitimate user are spoofed in order to pass authentication. A system is created in [6] to easily and quickly print finger prints capable of passing finger print scanner authentication. Similarly [8] describes how facial recognition may be tricked using an image of the legitimate user.

[3] discusses the vulnerability of various touch screen input methods to shoulder-surfing attacks while [4] measured the susceptibility of users to shoulder surfing attacks observing attackers having a ($> 20\%$) success rate on Android keyboards in the reproduction of a password after 3 attempts. Smudge attacks where, oily residues are used to detect common touch screen patters, also represent a significant security risk for secrets input via the touch screen. [5] explores the feasibility of such attacks finding in one experiment the lock screen pattern was partially identifiable 92% of the time and full identifiable 68% of the time.

These vulnerabilities have led to the development of lock screens which use a user biometric in an attempt to increase security. Facial recognition and finger print scanners have been employed as lock screen mechanisms. These methods insufficient to protect the device from attack. [6] demonstrates a method to exploit the finger print scanner on an android device using printed fingerprints. This method is simple and fast as long as a print of the authentic user's finger can be obtained. Other works by Germany's Chaos Computer Club [7] have show that it is possible to lift a finger print belonging to the authentic user from the device touch screen and use it to pass the finger print scanner authentication on Apple's Iphone $4S$ and 5. Attacks against facial recognition schemes involve spoofing the facial features of the authentic user in order to pass facial recognition authentication. [8] accesses the practically anti-spoofing measures in real word scenarios observing low generalization of and possible database bias in existing schemes.

In many situations, an attacker may not even need to pass the lock screen authentication. Imagine a situation where the authentic user of a device preforms the authentication, unlocking the device. The device then falls into the hands of an attacker. This attacker can not only exploit the authenticated state of the device, but may use the authenticated state to disable the lock screen entirely; Thus allowing permanent access to all applications on the device. [9] surveyed the lock screen behavior of 320 individuals ages $18-67$ with a median age of 31. Of these individuals, 39.6% rated themselves as having a very high level of IT expertise. The results of this survey indicate 56.3% of the participants did not use a lock screen at all.

Having a single authentication mechanism which verifies the user identity once when the device state changes from inactive to active has proven to be an ineffective method for delivering device security. The need for a more comprehensive security solution more is evident. We propose that a system which uses a biometric of user touch screen interaction to establish a user, device identity would be more comprehensive compared to solutions which are currently available.

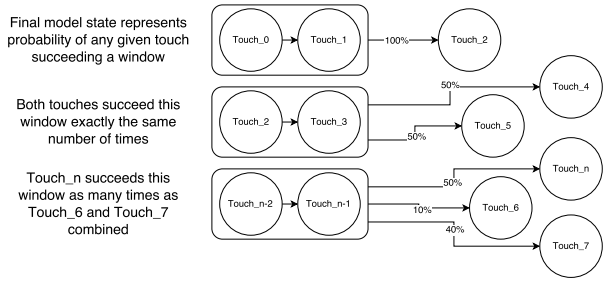

Figure 1: Example 2-Markov model after the probabilities have been calculated. The rounded rectangles indicate a sequence of touch interactions which precede the following touch interaction with some probability p. p for token T after sequence W is computed as occurrences of T after W by occurrences of W.

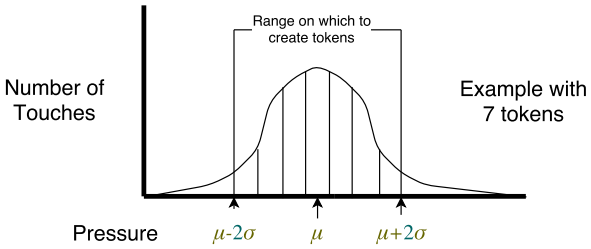

Figure 2: Tokens are only created for pressure range $\mu \pm 2\sigma$ for each key. Touches with pressure values $p > \mu + 2\sigma$ and $p < \mu - 2\sigma$ are thrown out. This eliminates outliers creating increased reproducibility.

Our solution is more comprehensive due to both the biometric nature of the data used and the operation throughout the use of the device.

3. THE SOLUTION

The main idea of this paper is that user touch screen interactions may be used to establish a user identity. This identity may be used in order to distinguish a legitimate user of a mobile device apart from illegitimate users. We implement a continuous authentication system based on user interactions with the soft keyboard of a mobile device. This system uses properties of the interactions including pressure and location to establish a user identity.

A large part of a user's interactions with a mobile device involve the input of data with a soft keyboard. These soft keyboard applications require that the user put their finger on the screen at a consistent location to indicate a given letter should be taken as input. This input is rich with information including pressure, key code, and timestamp.

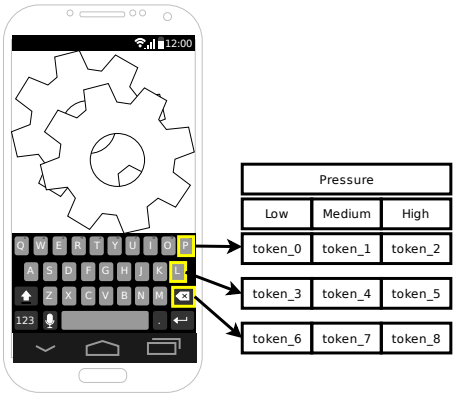

Figure 3: Multiple tokens, $k = 3$ above, correspond to each key location. A touch screen interaction is assigned a token based on key location and pressure.

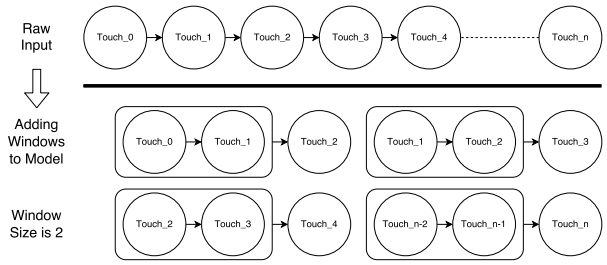

Figure 4: The top of this figure depicts the raw touchscreen interaction sequence. Each touch represents a single interaction between the human user and the soft keyboard. The diagram's lower portion demonstrates how the raw input is parsed into a 2-Markov model. The bottom left image can be interpreted to say that Touch_4 succeeds the sequence Touch_2, Touch_3 with some probability p.

As the keys on the soft keyboard are always in the same place on the screen, we take the key code value to represent the location in our model. Through interactions with multiple applications over time, the user produces a sequence of these inputs. This sequence is used to construct a model of user behavior.

The goal in creating the model is to be able to distinguish the behavior of one user from another user on the same device and from the same user on a different device. That is, the pairing of user and device will create a model unique to that pair. If either the user or device is changed, the model will be sufficiently different that it is detectable. The input sequence will be used to characterize the user-device pair. This uniqueness is possible because the pressure metric is a product of unique properties in the silicon of the device and the finger of the human user. Pressure is processed though the capacitive touch and sensor circuitry. The silicon's unique properties are a product of the fabrication process reflected when pressure is measured. The uniqueness of the human user is derived from the way in which they touch the screen.

Our Markov n-gram model calculates the probability of a given token following a specific token sequence of length n. Given a training sequence of tokens $T_0, T_1, \ldots, T_N$, we use maximum likelihood estimation (MLE) as follows to build the model. For all in-fixes of length n: $T_i, T_{i+1}, \ldots, T_{i+n-1}$, the following n-gram model is created: $P(T|T_{i..(i+n-1)}) = count(T, T_i, T_{i+1}, \ldots, T_{i+n-1}) / \sum_{T \in \Sigma} count(T, T_i, T_{i+1}, \ldots, T_{i+n-1}))$, where $T_{i..j}$ represents the token sequence $T_i, T_{i+1}, \ldots, T_j$. Here, we are computing the probability of next token being T given that the token sequence $T_i, T_{i+1}, \ldots, T_{i+n-1}$ has been seen. It is just the frequency of this event in the token sequence $T_0, T_1, \ldots, T_N$. $count(T, T_i, T_{i+1}, \ldots, T_{i+n-1})$ is given by the number of in-fixes with the same value as $T_i, T_{i+1}, \ldots, T_{i+n-1}$ followed by the token T. This gives $count(T, T_i, T_{i+1}, \ldots, T_{i+n-1}) = \sum_{j=0}^{N-n} (1 \text{ if } T_{j..(j+n-1)} == T_{i..(i+n-1)} \&\& T_{j+n} == T)$.

In the full Markov model, the probabilities for the state at time $t+1$ are determined by the state at time values 0 to t. In our implementation we use an n-gram Markov model which uses the previous n states to compute the next state probability. This n is referred to as the window size while the n states are referred to as a window. Each window precedes a single state. This approach develops a close approximation to the full Markov model while forgoing the complexity of computation associated with MLE. The result is an advantage in speed of computation.

The probability computation takes a sequence of previous states as input. Determining the probability for transitioning to another state given the current state can be done with the following algorithm.

1. compute the number of occurrences L for each window W
2. for all windows, for each token, compute $\frac{V_i}{L_i}$ where
   - V_i = number of token i which succeed window W_i
   - L_i = total number of occurrences of window W_i

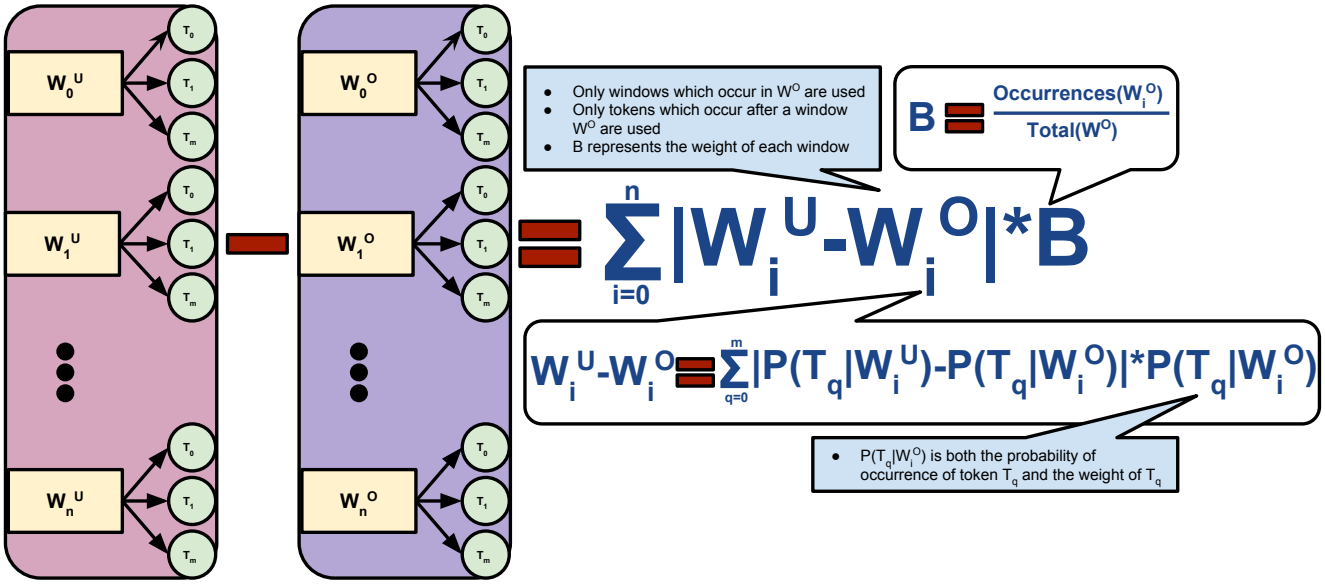

Figure 5: Description of the difference metric taken between W^U, the $n-gram$ Markov Model constructed from the authentic user touch screen interactions, and W^O, the $n-gram$ Markkov Model constructed from a different set of touch screen interactions. The goal is to determine how closely model W^O is to model W^U.

This computation is of less complexity than the probability computation for the full Markov model. In addition, there are only two quantities which need to be tracked. First it is necessary to know the total number of occurrences for a given window. Second the number of times a touch succeeds a window must be known.

Existence, in this case, is a binary decision. A person is either in a room or not in a room. Comparatively, touch interactions with keys have many degrees of complexity. A mobile device user is not simply touching a key or not touching a key, they are doing so by applying an amount of pressure to the screen within the area enclosed by the key. In this implementation we do not consider the variance in finger placement within a key, but we do capture pressure variations within a key. A state or token within the model is determined by the key pressed in combination with the pressure with which it was pressed. Under this methodology, all of the following would be considered different: "a" with pressure .5, "a" with pressure .8, and "b" with pressure .5. The first two illustrate a same key, different pressure scenario while the first and last describe a situation with different key, same pressure. In both cases our model takes these to be different tokens.

Our n-gram Markov model uses tokens which are a tuple of location and pressure generated through user touch screen interactions. There are a very large number of possible location, pressure combinations. Since it is unlikely that the user will be very precise in location or pressure when pressing a key. In other words, each key press, even from the same user on the same device, will not be exactly the same. If our implementation were to take all tokens to be different when location and pressure are not exactly the same, then there would be a very large number of tokens in the model. Having a large number of tokens creates a situation where each sequence of touch interactions will be different, even for the same user on the same device; this is not desirable. Further, if the entire space were used the variations in location and pressure, even when generated by a single user, would result in many n token sequences where are never produced more than once. This would say nothing about the user's affinity for producing certain token sequences. The user's tendency to produce recurring patterns is being used to classify the user's behavior. Therefore it is necessary to divide the available token space to accommodate the inconsistency of the user otherwise nothing can be said about a user's behavior.

We require a way of grouping a number of elements in this space together to make the sequences the user generates reproducible. Grouping here involves the selection of a range of values which will be considered equivilent. For instance, if there are apples of varying sizes which need to be categorized as "small", "medium" or "large", then we need to choose a range of values from some metric which could be used to describe all apples. Perhaps weight is chosen as this metric. Apples have far more than three possible weights. It needs to be determined which weights will be considered equivilent for purposes of categorization. For instance, the "small" grouping of apples might include apples which are less than 1.0 pound.

The goal in grouping elements is to increase reproducability of a particular pattern. In choosing how to group elements care must be taken not to group too many elements together. Variability within any given grouping is masked. If the uniqueness exhibited by two different users lies entirely within a grouping then it will be impossible to tell the users appart. To overview our method of grouping data, pressure and location values taken together form a tuple which represents a state or token. These tuples are considered equivilent if the location values fall within the same (x, y) coordinate

range and the pressure values fall within the same pressure range.

In our implementation, we choose to use the key code produced by a touch interaction as a representation for the location of the event. Given that users use a soft keyboard to input textual data, it stands to reason that interactions which produce the same key code have equivalent user intent. A different approach is used in determining what pressure values are to be grouped together. The two extremes for grouping the pressure metric are to consider all pressure values to be the same token on the one hand. In effect, this does not use pressure at all, equivilent to using only location. Alternatively all possible pressure values reported by the device could be used. The former masks any potential pressure variation unique to a user while the later captures the variation of a user at two fine a granularity to be reproducable. Neither scheme is entirely desirable, but both have desirable properties. Using all possible pressure values captures as much uniqueness as possible. This uniqueness would help to distinguish between users. However patterns created using this scheme would fail to be reproducible, even by the same user. Contrast this with grouping all pressure values together. This scheme produces patterns which are entirely reproducible, but fails to be able to distinguish between users. The goal is to maximize the degree to which users may be differentiated while maintaining reproducibility for the same user.

In our implementation, groupings for pressure values are selected based on the user's behavior. Pressure ranges are chosen around values which the user frequents. The goal of choosing the pressure component of the tokens in this way is to capture as much variation as possible. Consider an alternative scheme where ranges are uniform across all possible pressure values. It is feasible that the user uses a fairly consistent pressure when preforming all actions. Under uniform ranges this type of behavior might capture no variation; there is the potential for all pressure values to fall within one range. This is not desirable as only the magnitude of the pressure is captured. All of the potential variability contained in the pressure metric is lost. An alternative method of constructing the pressure ranges which does capture this variability is as follows.

1. Find the mean μ and standard deviation σ for all touch interactions' pressure values
2. Divide the range $[\mu - 2\sigma, \mu + 2\sigma]$ into k pressure ranges

k is then the number of tokens created for each location. The total number of tokens is k multiplied by the number of locations. 2 sigma is chosen because 95.45% of the user's pressure values will fall within this range [**?**]. This will throw away some touch interactions which have very high or very low pressures relative to the user's average. The benefit in doing this is that the pressure are then constructed around area where the user's variability is more likely to be expressed.

Windows within our n-gram Markov model are then a sequence of length n touch events. These sequences may overlap. To illustrate this suppose $n = 2$ and the user has input "apple". Each character has an associated pressure value, but suppose the number of tokens per location is 1. The effect of this will be that all equivalent characters, such as two "p" characters, will be considered to be the same token within the model. The windows for this sequence of characters will be ["ap", "pp", "pl", "le"].

Comparing models of user input is further complicated by the existence of multiple windows. Comparatively, the previous example described how to compute the average difference for a single window. The following algorithm outlines the computations completed to compute probabilities in our system. Suppose there are two lists of touch interactions which need to be compared. List U contains the authentic user's behavior while list O contains other touch interactions of unknown origin. The goal is to classify this list O as coming from the same user and device as U or not coming from the same user and device as U.

1. Compute the distribution (μ, σ) values for the the list U
2. Set the set the distribution of list O equal to that computed for U
3. For both lists, compute a set T of m tokens
   - One token is created for each location L
   - k tokens in the range $[\mu - 2\sigma, \mu + 2\sigma]$ are created for each location
   - $m = L * k$
4. For both lists, compute a set W of all windows
   - for each element in W, determine the number of occurrences
   - for each element in W, determine the successor tokens
   - let W^U represent the windows in U and W^O represent the windows in O
5. Determine the probabilities associated with a token succeeding a window as $P(T_q|W_j) = \frac{succeeds(T_q, W_j)}{occurrences(W_j)}$ where
   - T_q represents a token q from set T
   - $P(T_q|W)$ represents the probability of token T_q given that W_j precedes T_q
   - $occurrences(W_j)$ represents the number of occurrences of W_j in W
   - $succeeds(T_q, W_j)$ computes the number of times T_q succeeds W_j
6. Compute a weighted average of the difference between correspnding windows in W^O and W^U expressed as $\Sigma_i^n |W_i^U - W_i^O| * B$ where
   - n is the number of windows
   - W_i^U is the window in W^U which corresponds to and equivalent window W_i^O in W^O
   - B is the weight of W^O given by $\frac{occurrences(W^O)}{||W^O||)}$ where $||W^O||$ represents the total number of elements in set W^O

- $(-)$ represents the window difference operation
  - The window difference operation is a weighted average of the probability difference between corresponding successor tokens for a given W_i^O
  - expressed as $\Sigma_q^m |P(T_q|W_i^U) - P(T_q|W_i^O)| * P(T_q|W_i^O)$

There are several points from the above approach which require mention. The same distribution is used for both models. This is to ensure the tokens used when comparing models are the same. In computing windows, the previously computed set of tokens is used. If there is no window in W corresponding to a window in O, then the windows are considered to be maximally different from one another. In other words, having a window in list O which does not exist in list U is penalized by considering the the difference between the windows to be 1.0.

Such a system might be incorporated into the Android environment in the following way.

1. A background service could be used to collect `MotionEvent` objects from soft keyboard applications
   - touch interactions to be collected over time
   - a model of these user touch screen events could then be constructed in the background
2. Periodic authentication could compare new touch screen interactions against the existing model
   - a number of interactions would need to be collected before authentications can begin, discussed in Section 4
   - lower accuracy models can be construced with relatively few interactions
   - the number of touch screen interactions used in the authentication could be adjusted upward over time to achieve higher accuracies

The effect of such an implimentation is a small amount of added security initially which improves over time as more data is collected. Many actions could be taken if the result of this authentication finds the user is illegitimate. One approach might be to lock the phone, forcing the user to re-authenticate with some other method.

The following section provides evidence this scheme works. Support that this scheme is capable of using touch screen interactions to distinguish from among unique pairs of users and devices is provided.

4. THE DETAILS

In order to answer the question of whether or not touch screen interactions may be used in order to distinguish between users a system is implemented to gather and analyze data from interactions with the touchscreen of an Android device. This data consists of location, pressure, and time values associated with user's interactions with a soft keyboard. For gathering purposes, a keyboard application was

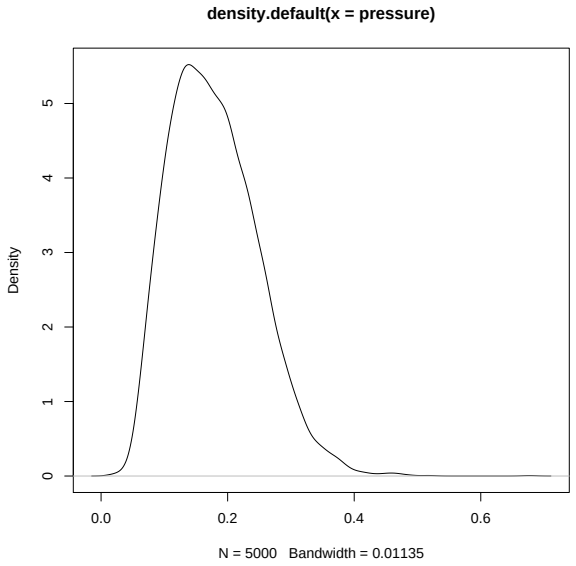

Figure 6: Density of touchscreen interactions' pressure values for one user.

modified to record the necessary data values. Two users were enlisted to acquire a large amount of data on two different Nexus 7 tablets. In effect this creates four user, device combinations. In order to analyze the collected data, the system described in 3 has been constructed.

The data were analyzed by

5. USER-DEVICE PAIR DISCRIMINATION

Some notable problems presented themselves throughout the course of this work which could influence the usability of such a system in a practical sense. The Android **MotionEvent** class **getPressure()** method used to collect pressure data in our system does not always return a high granularity of values. The pressure values are sometimes grouped into large steps. There might, for example, be 8 steps between 0.0 and 1.0 on some devices. This is a problem for our system which uses these pressure values to understand user variability. The problem seems to be device dependent suggesting that the device drivers may have a role to play in decreasing the reported graininess of pressure. Another possibility might be the values reported by the sensors, Perhaps these sensors only report a small number of steps.

6. RELATED WORK

In [10] a physical unclonable function (PUF) composed of a human biometric and silicon biometric leading to a unique user device identity. This work demonstrates how the PUF may be used for authentication. The user is presented with a polyline draw on the touch screen. The user then traces this line. The human pressure exerted in the trace and the exact traced path profile captures a biometric of the user. The silicon variability is extracted in a similar way to our implementation through capacitive touch and sensor circuitry of the mobile device. Our work differs in that we derive a model of user behavior over time which describes how the

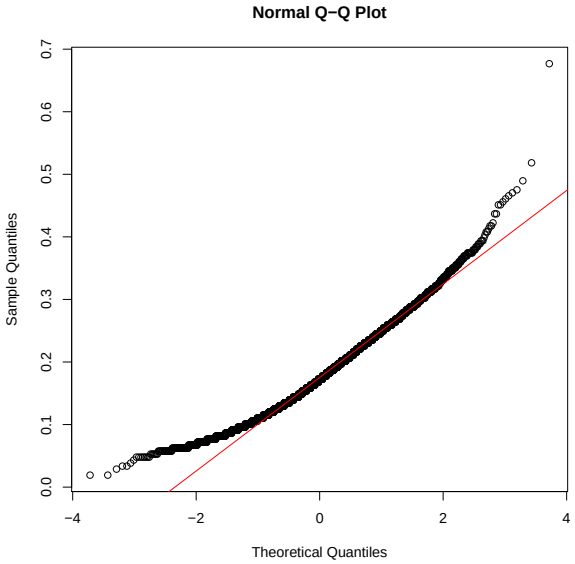

Figure 7: This Q-Q Plot describes data from a data set having a right skew compared to normally distributed data.

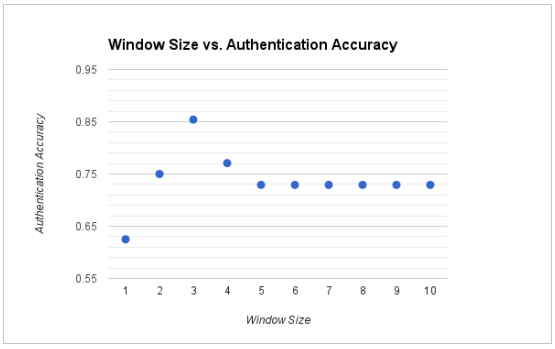

Figure 9: The effect of n in n-Markov Model window size on authentication accuracy.

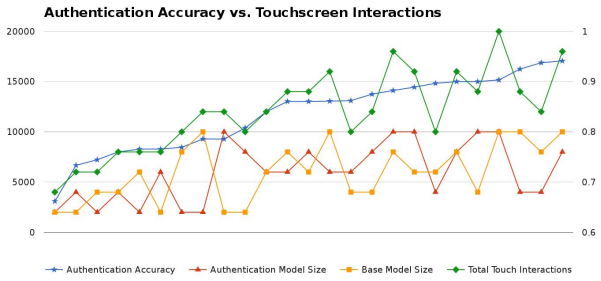

Figure 10: The left y axis describes the size of model in number of touchscreen interactions. We compare authentication accuracy(blue star), measured on the right y axis, to authentication model size(red triangle), base model size(yellow square), and total touch interactions(green diamond) measured on the left y axis. Total touch interactions is the sum of base model size and authentication model size.

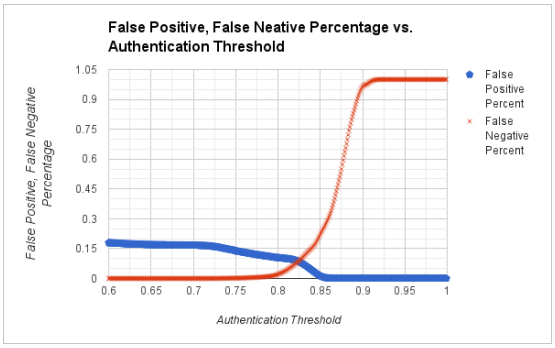

Figure 8: False positive and false negative percentages vary as the authentication threshold is adjusted.

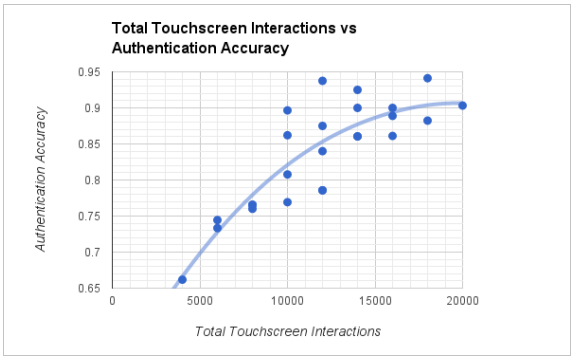

Figure 11: Authentication accuracy can be seen as a function of the total number of touch screen interactions used in model construction. Total touch interactions is the sum of base model size and authentication model size.

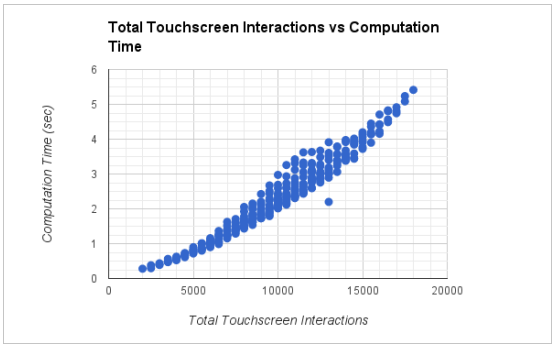

Figure 12: The computation time, measured in seconds(sec), on a Nexus 7 tablet is a function of $TotalSize = base\_model\_size + auth\_model\_size$. Model sizes are measured by number of touch interactions used in their construction.

user interacts with the soft keyboard. Comparatively [10] shows that, once trained for a particular challenge line, it is possible to differentiate pair of user, device from other pairs of user, device. In addition, we use discrete touchscreen events, key presses, while [10] uses continuous touchscreen events, swipes.

More good work done in the area of PUF's on mobile devices includes [11]. This work utilizes the silicon manufacture variability found in many different sensors combining them to create a model of the user. Again, the variability for this system also utilizes human biometrics. [11] uses data from the accelerometers, gyroscopes, and magnetometers to develop their model. By comparison, our system shows this may be done for touch screen interactions.

7. CONCLUDING REMARKS

The findings we have presented suggest that touch screen interactions may be used in order to distinguish a legitimate user of a mobile device apart from illegitimate users. Many current solutions utilize one-time authentication schemes These solutions do not provide sufficient protection in a mobile environment. This work represents a step toward a holistic approach toward mobile security, catering to threats existing in this environment.

This work presents as part of a positive trend in mobile security. Technologies part of this trend incorporate elements of the mobile environment, providing enhanced security by incorporating properties of the human user as part of the authentication. This approach is superior to exclusive use of things the user knows, because knowledge may be easily imitated while human biometrics can not.

8. REFERENCES

[1] Consumer Reports. smart phone thefts rose to 3.1 million in 2013 industry solution falls short, while legislative efforts to curb theft continue, may2014. `http://www.consumerreports.org/cro/news/2014/04/smart-phone-thefts-rose-to-3-1-million-last-year/index.htm`. Accessed: 2016-04-08.

[2] Federal Communications Commission. report of technological advisory council (tac) subcommittee on mobile device theft prevention (mdtp), december 2014, fcc. `http://transition.fcc.gov/bureaus/oet/tac/tacdocs/meeting12414/TAC-MDTP-Report-v1.0-FINAL-TAC-version.pdf`. Accessed: 2016-04-08.

[3] Florian Schaub, Ruben Deyhle, and Michael Weber. Password entry usability and shoulder surfing susceptibility on different smartphone platforms. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, page 13. ACM, 2012.

[4] Muhammad Daniel Hafiz, Abdul Hanan Abdullah, Norafida Ithnin, and Hazinah K Mammi. Towards identifying usability and security features of graphical password in knowledge based authentication technique. In *Modeling & Simulation, 2008. AICMS 08. Second Asia International Conference on*, pages 396–403. IEEE, 2008.

[5] Adam J Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M Smith. Smudge attacks on smartphone touch screens. *WOOT*, 10:1–7, 2010.

[6] Kai Cao and Anil K Jain. Hacking mobile phones using 2d printed fingerprints. 2016.

[7] Chaos Computer Club. instructions detailing how to create fake finger prints capable of passing finger print scanner authentication, october 2004, chaos computer club. `http://dasalte.ccc.de/biometrie/fingerabdruck_kopieren.en`. Accessed: 2016-04-09.

[8] Tiago de Freitas Pereira, André Anjos, José Mario De Martino, and Sébastien Marcel. Can face anti-spoofing countermeasures work in a real world scenario? In *Biometrics (ICB), 2013 International Conference on*, pages 1–8. IEEE, 2013.

[9] Marian Harbach, Emanuel von Zezschwitz, Andreas Fichtner, Alexander De Luca, and Matthew Smith. ItâĂŹsa hard lock life: A field study of smartphone (un) locking behavior and risk perception. In *Symposium On Usable Privacy and Security (SOUPS 2014)*, pages 213–230, 2014.

[10] R. Scheel and A. Tyagi. Characterizing composite user-device touchscreen physical unclonable functions (pufs) for mobile device authentication. In *ACM International Workshop in Trusted Embedded Devices, TRUSTED 2015*. ACM, October 2015.

[11] Joy Zhang, Xiao Wang, Pang Wu, and Jiang Zhu. Sensec: Mobile security through passive sensing. *2013 International Conference on Computing, Networking and Communications (ICNC)*, 0:1128–1133, 2013.