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. Often times these attackers go through the trouble of preforming these attacks because gaining access to the data might be quite lucrative. Take the example of

Many current practices keep mobile devices secure utilizing some nature of 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 a large amount applications, tools, and data on the device. Some of this data could be potentially damaging if compromised by a malicious party.

One issue with this nature of user verification There are many ways an attacker might trick this one-time authentication hence gaining access to resources on the device.

This paper focuses on an idea for creating added security for data stored on mobile devices. Often times there exists a trade off between security and convenience from the user's perspective. This can be seen in solutions such as where The solution presented in this paper provides additional security while requiring no additional actions from the user. This is accomplished by recording touch screen interactions in the background while the user interacts as they normally would with the applications on their mobile device. The most resent interactions are then tested against previous actions to determine if the user's pattern has changed.

We provide a step forward, enhancing the ability of mobile devices to recognize when the device may have been compromised. Compromised here meaning the device is in 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 user's pressure when touching the screen and the location on the screen being touched 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 both the device and the user. Section 4 establishes that a difference in either device or user may be detected by our implementation.
- There is no convenience cost assessed upon the user. All security improvements are accomplished without requiring any additional actions from the user. The specifications of the implementation used to accomplish this transparency are discussed in Section 3
- The performance of this system is sufficient to make it practical. A performance comparison is presented in Section 4 which demonstrates the running time of the system on a Nexus 7 tablet.
- We improve upon the ideas discussed in ? utilizing similar concepts for modeling interactions. We extend beyond this work, modeling soft keyboard interactions to describe user behavior over time. Section 5 provides further discussion of ?with emphasis on similarities and differences between work described throughout this paper.

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. A type Markov model which uses n previous states in computing next state probabilities is used for the purpose of developing these probabilities. This model is explained more throughly in Section 3.

2. THE PROBLEM

3. THE SOLUTION

The main idea of this paper is that user touch screen interactions 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.

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. 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. The silicon's unique properties are a product of the fabrication process while the uniqueness of the human user is derived from the way in which they touch the screen.

Let's say we want to model the behavior of an individual in terms of where they choose to spend their time. Say further that the goal in creating this model is to predict their $t+1^{st}$ location based on their current location. The outcome of using a Markov model to describe a system is a vector $\hat{P}$ of probabilities for each possible state. For the individual in our behavior model, $\hat{P}_i$ corresponds to the probability that the i_{th} location will be the location of the person at time $t+1$ if the current time is t. Such a probability outcome for our individual might be that if at time t the person is in the living room, then at time $t+1$ there might be a 70% probability they are still in the living room, a 20% probability they are in the kitchen, and a 10% probability they are in the bathroom. Such a model might be useful in understanding the behavior patterns of one person, or comparing behavior patterns among persons.

Continuing the example of predicting the next location of a person, we stated the outcome of the model as a vector giving probabilities for each state. The full Markov model uses the sequence of all previous states in computing the probability of the next state. This approach is grounded in maximal likelihood estimation(MLE). MLE is formulated as follows. Each probability depends on all previous states of the system according to :::

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. The window precedes a single state. The location of the person in the previous example might be predicted by the previous three rooms they located themselves within. In other words, location $t+1$ might be predicted states $t-2$, $t-1$, and t 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

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.

Our implementation utilizes an n-gram model. Similar to the earlier example of predicting locations, our goal is to predict a user's behavior in interacting with a soft keyboard. Locations are analogous to keys, but an interaction with a key on the soft keyboard is more complex than existing in

a room. 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 states.

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, having such a large number of tokens creates a situation where each sequence of touch interactions will be different for the same user on the same device; this is not desirable. In fact, if the entire space were used the variations in location and pressure, even when generated by a single user, would result in a very large number of tokens each having low probabilities of succeeding any n token sequence.

We require some way of grouping numbers of elements this space to make the sequences the user's generate reproducible. Grouping here involves the selection of a range of values which will be considered the same. 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. The "small" grouping of apples might include apples which are less than 1.0 pound. In choosing how to group elements care must be taken not to group too many pressures and locations together; doing so would fail to capture characteristics of the sequence unique to a user. Thus multiple users would present with similar sequences making it difficult to distinguish among users.

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. 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 would capture the variation of a user at two fine a granularity. Neither scheme is are desirable as a whole, 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 patters created using this scheme would fail to be reproducible by the same user. Contrast this with grouping all pressure values together. This scheme produces patters 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 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 tokens created for each location. The total number of tokens is k multiplied by the number of locations. 2 sigma is chosen because 95% 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"].

Let us return to the situation where a Markov model was used to predict the $t + 1^{st}$ location of an individual. Now suppose that these probabilities have been developed for two people, A and B, who's behavior patterns we would like to compare. Our goal is to quantify the difference between the location patterns of these individuals. Suppose that in both cases there are three rooms, living room, kitchen, and bathroom, and that the locations of A and B are measured at 4 time instants. A is in the living room during the first time instant. In subsequent time instants, A is in the kitchen then living room then bathroom. The result of these movements is the probabilities 0%, 50%, 50%. These probabilities describe A's how likely A is to be in the living room, kitchen, and bathroom at the next time instant, $t + 1$, if A

is currently in the living room. B, on the other hand, at the measured time points is located in the living room then living room then kitchen then bathroom. In this case the result is probabilities 50%, 50%, 0% that B will be in the living room, kitchen, and bathroom at the next time instant if B is currently in the living room.

Now suppose the goal is to compare the movement of A and B from the living room. The end result should be some number which represents the difference in probabilities between A and B. Our approach consists of computing the average absolute difference between the probabilities. In the scenario described above the average difference between A and B is computed by $\frac{|0\%-50\%|+|50\%-50\%|+|50\%-0\%|}{3}$. The resulting value, 0.33, is a floating point value between 0.0 and 1.0 which represents the closeness of the two vectors. 0 indicates maximal closeness while 1 describes two vectors which are a different as possible.

In our implementation the difference between two models is computed with a method analogous to the comparison of movement between A an B. 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 touches of unknown origin. The goal is to classify this list O as coming from the same user an 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 k tokens in the range $[\mu - 2\sigma, \mu + 2\sigma]$
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 touches
   - 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 $P(T_i|W_j) = \frac{succeeds(T_i,W_j)}{occurrences(W_j)}$ where
   - T_i represents a token i from set T
   - $P(T_i|W)$ represents the probability of token T_i given that W_j precedes T_i
   - $occurrences(W_j)$ represents the number of occurrences of W_j in W
   - $succeeds(T_i, W_j)$ computes the number of times T_i succeeds W_j
6. For each W^O, compute the difference between the corresponding W^U.
   - The difference between windows is defined as the average difference between the successor touch probabilities
7. The difference between the models is then $\Sigma_i^n \frac{|W_i^U - W_i^O|}{||W^O||}$ 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
   - $-$ represents the window difference operation described above
   - $||W^O||$ is the number of windows in 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. A background service could be used to collect MotionEvent objects from soft keyboard applications. This would allow touch interactions to be collected over time. A a model of these touch events could then be constructed in the background. Periodic authentications could compare new touch interactions against existing older touches which have come from the user. Many things could be done if the result of this authentication find 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

5. RELATED WORK

6. 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.

[1]

7. REFERENCES

[1] Rosenfeld, K., Gavas, E., and Karri, R. Sensor physical unclonable functions. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on* (2010), IEEE, pp. 112–117.