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

A mobile device user continually interacts with many sensors through the natural user interface (UI) of apps. These interactions are unique for each (user, device) pair forming a user-device biometric. A physical unclonable function (PUF) can be realized from the touch screen pressure variability. We illustrate how a sequence of these pressure values from discrete touchscreen interactions may be used to uniquely characterize a user-device pair. These touch screen interactions Markov models can be integrated into a continuous authentication layer. Based on the most recent sequence of touch screen interactions, the continuous authentication layer can assign a probability that these interactions came from the authenticated (user, device) pair. Continuous authentication may help protect access to a mobile device from a malicious party by detecting the anomalies early. The effectiveness of this scheme is described in terms of how definitively can one user be differentiated from another.

1. INTRODUCTION

Mobile devices are ubiquitous in the modern world. These devices are becoming progressively more important for many applications with hold sensitive data such as financial and health care data. Securing mobile devices poses unique challenges and opportunities. Traditional user authentication is a single challenge-response model based on a password. A mobile device, however, has a greater number of available device sensors that have been used to capture the uniqueness of a specific user, device interaction for enhanced traditional user authentication. Traditional user authentication however leaves large gaps in between successive user authentications. If the device is compromised in between two successive user authentications, there is a long time available to the adversary for inflicting damage. This paper proposes a continuous authentication framework in which an authentication framework running in the background is continually monitoring user activities. Moreover, this monitoring is based on the combined biometric of the user and the device silicon captured as a physical unclonable function (PUF).

Traditional physical unclonable functions (PUFs) which generate a unique signature of a given hardware device are not sufficient to guarantee the identity of a user. A mechanism that combines the biometric identity of a human user and the device silicon in a way that is not mathematically separable is more robust in the mobile world. UD-PUFs (user-device entangled physical unclonable functions) [**?**] serve just that role. Integration of such UD-PUFs into a continuous authentication environment is the main problem addressed in this paper.

A UD-PUF is a function of both the device silicon and the user behavior. Such a function must show significant change in the output given a small change in any component of a user-device pair somewhat like a hash function. Hence the UD-PUF must be based on a property or properties which vary significantly and identifiably among mobile device hardware and users. The best candidates to use will be properties which offer the most variability, and properties which are most easily exposed in the android operating system.

System efficiency and response times imposes additional constraints on properties to base a UD-PUF on. The system must also be practical under normal use conditions for mobile devices. The system must be non-intrusive to the user, fast enough to run on mobile devices, and accurate when authenticating users. If a user must spend a long time authenticating they are not able to use this time to accomplish the task for which they are being authenticated. This detracts from the efficiency of using a mobile device to complete the task. If the proposed system is too computationally intensive to be run on a mobile device, or the system has low accuracy then it is useless for any practical application.

In the continuation authentication world, a composite sensor vector [3] has been used to establish a user identity. Other user biometrics such as inter key stroke timing have also been used. In this paper, the continuous authentication is based on a combined user-device identity. The continuous authentication signature will fail if either the biometrically correct human being or the biometrically correct device component is removed. This paper builds the continuous authentication framework on the user touch screen interactions. We establish that the user touch screen interactions do differentiate one user from another. Note that there are advantages to basing a framework on a single source of information. A failure of the touchscreen to function also constitutes device failure. This system is more robust because it will only fail upon failure of the device touch screen. Compare this to other systems whose functionality depends on many components. A failure in any one of these components disrupts the authentication scheme making these systems more prone to failure.

Continuous authentication frameworks' basis premise is that a user behavior over time gravitates towards predictable. It can be frequently modeled as an n-Markov model. It states that the user tokens of length n repeat themselves with certain frequency. Hence if we can record history of n-token sequences, they could help us classify the user's current behavior. The tokens are touch screen pressure values that are continually generated as the user interacts with an app through a soft keyboard.

Consider an authentication scheme which seeks to achieve authentication over the duration of a device's use, but requires that some action must actively be taken by the user. Say that this system requires the user to complete some gesture in response to an image on the screen. This sort of authentication my interupt the user durring important tasks, or otherwise cause the device to be more difficult to use. This may lead to great frustration on the part of the user at having to reauthenticate. This senerio exposes a clear benefit of continious authentication which runs in the background. If the authentication system utilizes actions which are already being preformed by the user to preform the authentication then the user can go about their business completly unaware and uninterupted by the authentication system. This means additional security is possible without causing undo stress on the user or making interation with the device an unpleasent experience.

The structure of the paper is as follows. Section 2 discusses related work. Touchscreen pressure and the corresponding UD-PUF model are described in section 3. The n-Markov model and its parametrization are discussed in section 4. Data collection methods are discussed in section 6. The authentication scheme is discussed in section 5. The results including final numbers and suggestions in practical applications are articulated in section 7. Conclusions are presented in section 8. Finally section 9 discusses future work in this area.

2. RELATED WORK

SenSec, a similar authentication scheme to the one proposed in this paper, completed at Carnegie Mellon used information from from accelerometers, gyroscopes and magnetometers to construct a model of a user. [3] This method can be classified as a UD-PUF as each accelerometer, gyroscope and magnetometer will have some variance inherent in the manufacturing process for these devices [**?**]; the input to these devices will also vary significantly by user [**?**]. As a result the output is a function which varies with changes in user or device, a UD-PUF.

Another noteable aspect of the SenSec system is their method of modeling users with an n-Markov model. This system presents several benefits which include relative simplicity and scaleability. [3]

[2] [1]

3. TOUCHSCREEN PRESSURE

Many mobile devices today are equiped with capacitive touchscreens which have the ability to approximate, among other metrics, the pressure of a given interaction with the touchscreen. [**?**] When a user interacts with the touchscreen of their device they typically put their finger or fingers on the screen in order to indicate what operation they want the device to preform. This contact (or near contact) functions to increase the capacitance of the screen in a predictable way. The touchscreen is split into many rows and columns; capacitance is measured indirectally as a function of the current increase in the row and column containing the touch.**??**

If our goal is be be able to distinguish a user's interaction with a given device from this same user on a different device and from different users on any device than our description of the user will need to be a product of both the user and the device. In the android operating system there exists a pressure function which returns a value proportional to current at sides of the touchscreen for a given touchscreen interaction. [**?**] This measurement is a function of both the device silicon and the way the interacts with the touchscreen making this measurement a reasonable choice for use in our system. This is the value which will serve as the basis for our scheme and will henseforth be referred to as touch pressure.

The pressure function is significant because it's value not only depends on the characteristics of the device but also on the way in which a user interacts with a given device. The effect of a given device on the touch pressure value will differ significantly due to variations implicit in the manufacturing process for the touchscreens of these devices. [**?**] Our supposition is a given user will interact with a touchscreen in such a way as to cause significant variations in the touch pressure values when compared to other users on the same touchscreen [**?**]. Given this, we have chosen touch pressure as the basis for our UD-PUF.

In our system, we reccord the touch pressures generated by the user when they are using the system's soft keyboard. Once we have collected a sufficient number of touches, authentications may begin to be preformed on new data. Figure 6 demonstrates that as few as 6000-8000 touches may be used to achieve accuracies higher than 80%. A typical user can enter information into a devices soft keyboard at a rate of xx words per minute. **??**. A word is on average five letters long leading to a rate of xx*5 touch interations per minute. This means in the average use case a user will generate enough information to begin being authenticated in about x minutes of continious use. Many applications utilize this as a method of input meaning that in the average use case there will be a constant influx of new data to be analyzed.

4. MODELING A USER-DEVICE PAIR

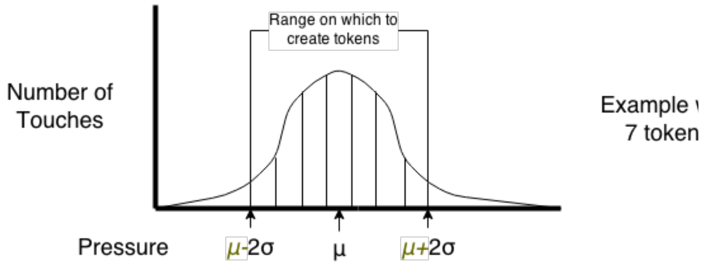

Figure 1: Describes how a distribution may be split into distinct token ranges by pressure.

Interactions between users and devices are complex. To interpret these actions in a meaningful way, in order to perform an authentication for example, it is necessary to simplify these interactions. The chosen model must provide sufficient entropy such that a model generated with a given user-device pair is not consistently reproducible by another user or on a different device. The modeling method must also be easily reproducible by the original user on the original device. A model having the necessary characteristics required for this application is a Markov Chain.

Markov Chains are useful in predicting systems whose behavior can be modeled in discrete states. The transitions between states can be identified to happen with some probability. In general an n-Markov model states that the user tokens of length n repeat themselves with certain frequency. Hence if we can record history of n-token sequences, they could help us classify the user's current behavior. The tokens are touch screen pressure values that are continually generated as the user interacts with an app through a soft keyboard.

Upon identification of an appropriate model the next step is to discover an optimal way in which it may be applied to the current problem. Interactions between a user and device can be described as a sequence of touch pressure values which accumulates over time as the user continues to interact with apps though a soft keyboard. We use n-sized subsets of this sequence in the n-Markov model constructed for the user.[**?**]

In building the model we remove some touches likely to be mistakes by the user or simply outliers in the data set. The distribution of touch pressue values is calculated for each area on the touchscreen with which the user interacts. In our system these areas correspond to keys of the soft keyboard but they are not restricted to be this. The distribution, μ and σ, values are calculated for each of these areas. If a key falls outside of $\mu \pm 2\sigma$ for a given area, then it is not included in any of the n-token sequences. Figure 1 illustrates how this works on a distribution which has been split into 7 token ranges. The value of $\mu \pm 2\sigma$ was chosen because statistically xx% of touches will fall withing this range for normally distributed data. **??** Figure 2 plots a set of touch pressures from one of our users which shows that our data can be described as having a normal distribution. For tokens to be considered equal, both the area from which the touch was generated and the pressure must match.

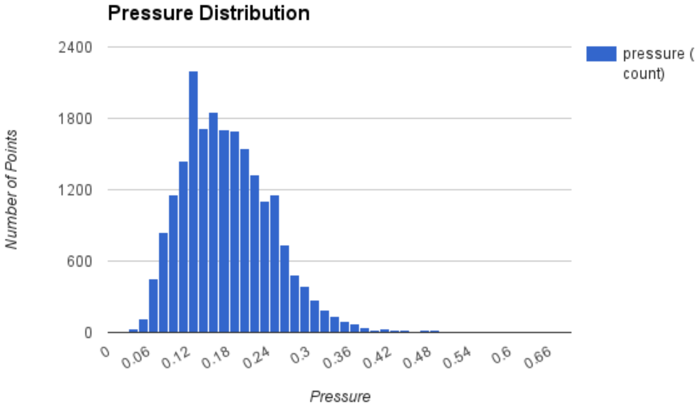

Figure 2: This chart displays a set of pressure data from one of our uses. Note that the pressure values are nomally distributed.

The goal in modeling a system with a Markov model is to classify the system in terms of its transitions between states. If such a model is to be used to purposes of uniquely identifying a given system, than the model must be chosen in a way which exposes the uniqueness of the system. Our scheme uses a Markov model constructed from the sequence of touches entered by a user through the soft keyboard. Each touch contains information about the pressure and location of the interaction with the touchscreen.

Our Markov model calculates the probability of a given touch coming after a sequence of n touchs. n is a parameter to the model which should allow for increased accuracy in determining the true probability of a given touch to come after a preceeding sequence of touches when given more data to construct the model. The increased accuracy is due to a greater number of possible model states which decreases the liklihood that a user will have states which overlap a different user. Figure 3 demonstrates how the sequences for the Marcov model are generated from the user's raw input into the soft keyboard.

The probability of a touch coming after a sequence of touches can be expressed as the number of occurrences of touch after a given sequence by the total number of occurrences of that sequence. The idea behind this probability calculation is illustrated in Figure 4. Noteably, Touch_n is not distinct. In other words Touch_a will be considered equal to Touch_b if the keycodes of these touches are equal and the touches fall within the same pressure range. Pressure ranges are depicted in Figure 1.

The information needed to calculate the probability of a touch succeeding a given sequence is the number of times that touch succeeds the sequence and the number of times the sequence occurrs. The use of a prefix tree as a data structure can be used to make both of these numbers more readily accessible thus increasing the speed of the probability calculation.

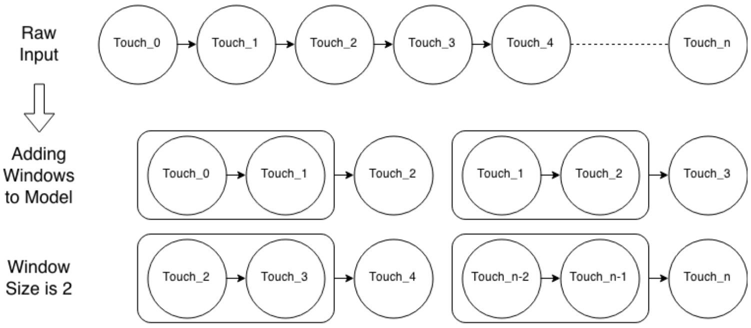

Figure 3: The top of this figure depicts the raw imput origionating from the touchscreen. Each touch represents a single interaction between the user and the human user and the soft keyboard of the mobile device. The bottom images show how the raw input is organized into the marcov model. For example, the bottom left image can be interpreted as Touch_4 succeeds the sequence Touch_2, Touch_3 with some probability.

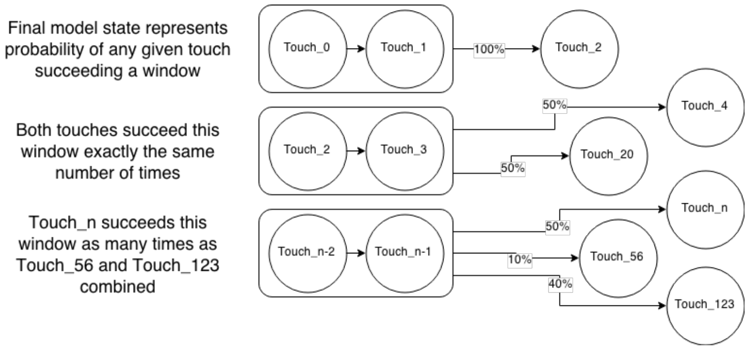
Final model state represents probability of any given touch succeeding a window
Touch_0
Touch_1
100%
Touch_2
Both touches succeed this window exactly the same number of times
Touch_2
Touch_3
50%
Touch_4
50%
Touch_20
Touch_n succeeds this window as many times as Touch_56 and Touch_123 combined
Touch_n-2
Touch_n-1
50%
Touch_n
10%
Touch_56
40%
Touch_123

Figure 4: This figure depicts the state of the Markov model after the probabilities have been calculated. The top image, for example, may be interpreted as Touch_2 succeeds the sequence Touch_0, Touch_1 with probability 100%.

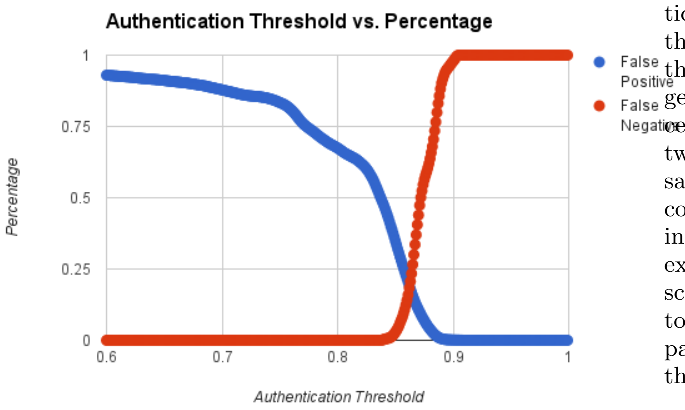

Figure 5: Describes how false positive and false negative percentages vary as the authentication threshold is adjusted.

In our system the n-token sequences are stored in a list while a prefix tree is used to store information about the each sequence. This information is stored on the nodes of the tree; it includes the number of occurrences of a sequence and a list of indexes where the sequence occurrs in the list storeing all the sequences. This index list is useful because it eliminates the need to search the list in order to determine what the successor touches of a window are. This approach requires much fewer calculations compared to simply using a list to store the sequences. If only a list is used then in order to determine how many times a sequence occurrs the list must be searched.

??

5. DIFFERENTIATING USER-DEVICE PAIRS

In distinguishing a particular user from another different user, it is necessary to develop a method of comparison between users. In our method of comparison we take the probability associated with a touch pressure coming after a sequence of preceding touch pressures for a particular user and compute the difference between this probability and the probability of the same touch pressure coming after the same sequence of touches for a different user. The average of these probability differences is taken to be the difference between two users. Once a comparison is established a natural extension is a system of authentication. This system needs to determine when two sets of touch pressure values came from the same user-device pair. When authenticating a user, we take one minus the average difference between the model constructed from the two sets of touch pressure values. Take this value to be the authentication percentage for a given set of touch pressure values against another. To determine how well this system does at differentiating users it is useful to develop metrics which describe the system's performance under conditions which are similar to it's potential real-world applications. Figure 5 illustrates how false positive percentage and false negative percentage vary based on where the threshold for authentication is set.

Here, authentication threshold refers to the value of authentication percentage one model must have against another for the models to be considered the same; two models which are the same are supposed to have been created from touches generated by the same user-device pair. False positive percentage measures what fraction of authentications between two sets of touch pressures which did not come from the same user-device pair, therefore these sets should not be considered the same, but did authenticate as being the same in our authentication system. False negative percentage is exactly the inverse of false positive percentage in that it describes what fraction of authentications between two sets of touch pressures which did come from the same user-device pair, but did not authenticate as being the same in our authentication system.

In Figure 5 there exists a clear intersection between false negative and false positive percentages. This intersection is significant; at this point the system neither biased toward allowing user-device pairs which should not authenticate to pass authentication nor toward disallowing user-device pairs, which should authenticate, from passing authentication. This point represents a balance in design, but the best authentication threshold will depend on the application of this system.

The implementation of the authentication system is as follows. A Markov model is constructed from a sequence of touches known to have come from the user. A seperate Markov model is then constructed from a sequence of touches which need to be compaired to the model. The model coming from the later sequence of touches is created with the same distribution as the base model. This affects the n-sized touches sequences which are added to the authentication model. Sequences containing touches which fall outside the base model's distribution are thrown out. The probabilities computed for each of these models are then compaired and a percent difference is derived from these compairasons. We then choose to authentication only those models which have achieved a low enough percent difference.

In comparing the base and auth Markov models each n-token sequence in the auth model is compared to the base model. For a given sequence in the auth model the base model is searched for a matching sequence. If a matching sequence is not found then the sequence in the auth model being considered is decided to be maximally different. If a matching sequence is found than the absolute value of the differance between probabilities computed for that sequence are considered to be the difference. The aggragate difference is the average of these difference values.

6. DATA COLLECTION AND ANALYSIS

Data for creating touch pressure models in this experiment was generated using a special keyboard application for the android operating system. Users would load the keyboard onto their device and continue using the device in the way they would normally. Some users were asked to play a typing game in order to help expedite the data collection process. After the users had generated at least ten thousand touches the data was collected from the user's device.

The results presented here were derived from four user-device

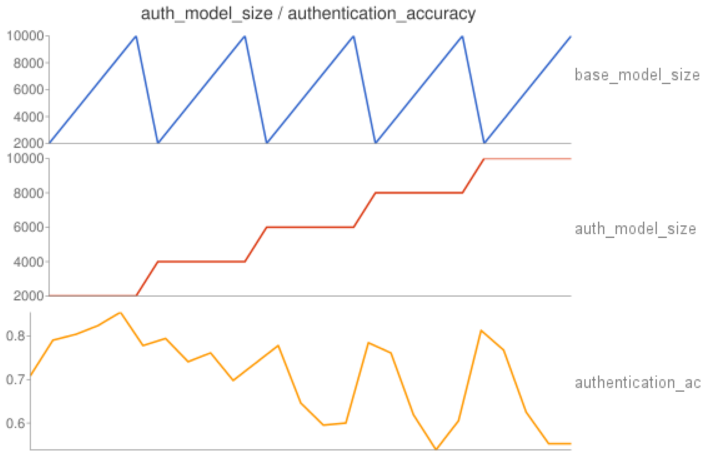

Figure 6: Authentication accuracy is a function of both the base model size and authentication model size.

pairs each having a large number of touches. Each user was asked to use two different devices to collect data. The size of the data sets collected was approximately 20000-25000 for each user-device pair.

7. RESULTS

We predicted that an increase in the number of touches used to build the model would provide better accuracy in authentications. In additon,

In exploring the design space of this system we are trying to determine how well the system will perform in the end use case. Perhaps the best measure of the system's performance is how accurate the system is when authenticating users. Figure 6 depicts authentication_accuracy as a function of base_model_size, number of touches known to have originated from an authentic user, and auth_model_size, number of touches which are to be checked against the model generated from the base touches. We define authentication_accuracy to be the percentage of authentications for which our system makes the correct decision. In other words, an authentic user is authenticated and a non-authentic user is not authenticated. The size of base model and user model which result in a given authentication accuracy are aligned with that authentication accuracy on the horizontal axis in the chart.

In some instances, increased numbers of touches did not result in higher authentication accuracies. These lower accuracies seem to correspond to areas where This could be a result of the way the model decides to include n-sized touch sequences in the authentication. In this case, an auth model which is different will authenticate with low probability because sequences are punished for not existing in the base model if they exist in the auth model. If there are not many n-token sequences in the base model then high differences will be represent for models which are only slightly different, because many sequnces will not exist in the base model. This sequences will be considered maximally different and constitute a high proportion of the number of n-token sequences.

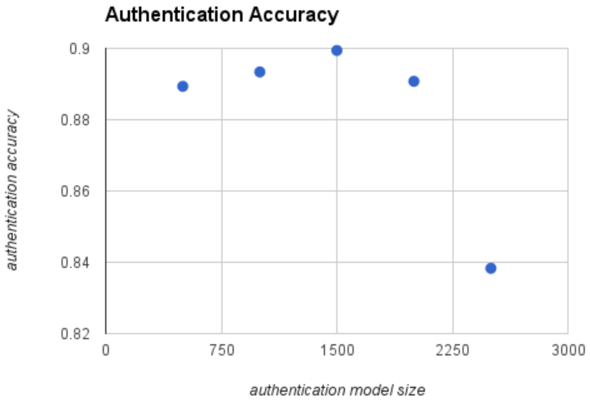

Figure 7: Depicts the result of many model comparisons done around the area of best results in Figure 6.

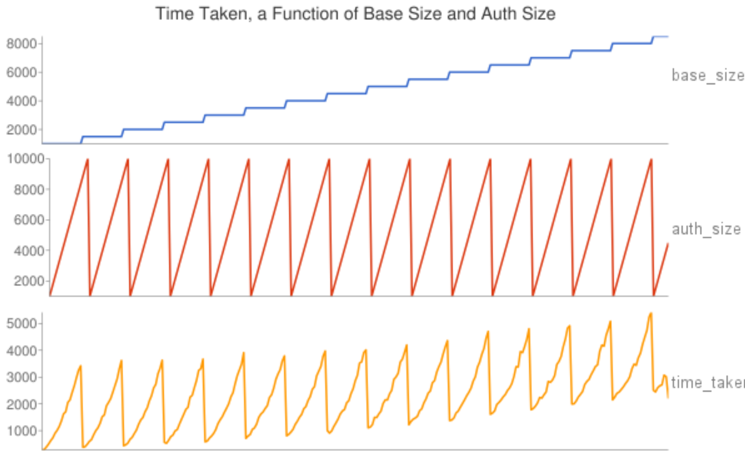

Figure 8: Depicts the time taken on a Nexus 7 tablet as a function of base model size and auth model size. Time taken is measured in milliseconds while model sizes are measured by number of touches.

To establish that the results in Figure 6 hold for large numbers of comparisons many more comparisons where done around the area of best results. The results of these tests are presented in Figure **??**. For the results depicted in the figure, user_model_size is held constant at ten thousand while auth_model_size is varied. The test was performed in this way because variations in auth_model_size seems to have a greater impact on authentication accuracy than variations in base_model_size. Approximately the same trend as seen in Figure 6 presents itself in Figure 7.

Figure 8 depicts the runtime of our system on a Nexus 7 tablet. Time taken is measured in milliseconds while model sizes are measuered by number of touches used to construct the model. The time taken metric does not include the overhead associated with adding touches to either the base or auth models; it is assumed this will be done over time as the user enters data. In addition, adding touches is not a

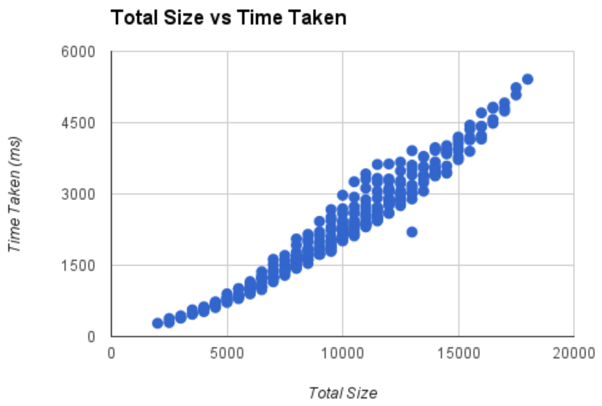

Figure 9: Depicts the time taken on a Nexus 7 tablet as a function of base model size plus auth model size. Time taken is measured in milliseconds while model sizes are measured by number of touches.

computationally intense activty as it is designed to be done in constant time. Time taken does include the probability computation for each of the models and the compairason between the models. This chart is a good representation of how each of the model sizes affect the runtime of the system and allows for the identification of an overall trend in the system's runtime. The overall time taken is trending upward as both base model size and auth model size increase; this suggests that the total number of touches used, that is the sum of base model size and auth model size, might be the most important factor in determining the runtime. The dependence on the total number of touches makes sence because the majority of the computations are used to determine the probabilitys of the Marcov model. A greater number of touches in a model reqires more of these probility computations to take place. Also of note, base model size and auth model size seem to affect time taken differently. A change of k in the size of auth model seems to increase the total amount of time taken more than the same k-sized change in the base model.

Figure 9 illustrates how the time taken depends on the total number of touches used in createing the models. That is to say total size is the sum of base model size and auth model size. The trend in the model suggests that the total amount of time taken will increase exponentially as the number of touches used to generate the model increase. This figure also supports the conclusion that the total size of the model in number of touches has a larger influence on the runtime then the sizes of either the base or auth models individually; in general, more touches will lead to an increase in run time.

Figures 6 and 7 establish that in general a greater number of touches used in the authentication will result in a greater accuracy. This manifests in the charts as the peaks indicating the highest authentication accuracy existing aground areas where the largest numbers of touches are used to construct the models. Figures 8 and 9 demonstrate the performance tradeoff associated with using increased numbers of touches. As expected there exists an inverse relationship between performance in terms of speed and accuracy of authentication. That is, increased authentication accuracy comes at the expense of execution speed.

8. CONCLUSIONS

This paper presents an approach toward continuous authentication which utilizes the variability in the way users interact with the touchscreens of their devices to differentiate distinct user-device combinations. This is useful in situations similar to theft where a mobile device is accessible to a party which may desire to use the device to compromose sensitive data. One shot authentication systems like passwords are not the idea solution in this case. If the password has also been compromised or is weak such that it may be easily broken, then the attacker has access to the device indeffonatly. In the continuious authentication model presented here, if the device is compromised then the way the attacker interacts with the device will deviate away from the model which has been established for the authentic user. After the the newest sequence of touch interactions has become significantly different from the established model, then the device may be locked or some other arbitrary action taken.

The implementation of our system has both modeling and authentication components. The modeling component depends on the use of an n-Markov model constructed from a sequence of touches entered by the user. This allows for the likelihood that two sequences of touch pressure values came from the same user to be calculated in a probabilistic fashion. The authentication system constructs two models. One of the models coming from the most recent touches and the other coming from touches preceeding the most recent. A differance is derived between the models which is a function of the probabilily derived in the modeling phase. If differance is sufficiently large it can be said with reasonable certainty that the user from which the touch sequences origionated for the most recent touches differs from the user who generated the touches preceeding those most recent.

Data for this approach comes from the user's interactions with the soft keyboard of their device. This data will be generated over time by the user; thus this scheme lends itself very well to a continuous authentication model. In addition, this data is reliable over the lifespan of the device. The authentication system will persist as long as the touchscreen functions correctly. A failure of the touchscreen renders the mobile device useless; hense the system is more robust compared to systems which depend on multiple sensors contained on the mobile device. Mobile device touchscreen interactions are also ideal for use in an authentication scheme. They function both as a biometric of the human user and the silicon of the device; a change in either device or user will be detected as a result.

Depending on the implementation of this system, varying the parameters of the modeling system can allow the implementer to tune the system to their specific purpose.

9. FUTURE WORK

10. REFERENCES

[1] T. Feng, Z. Liu, K.-A. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. K. Nguyen. Continuous mobile

authentication using touchscreen gestures. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pages 451–456. IEEE, 2012.

[2] W. Shi, F. Yang, Y. Jiang, F. Yang, and Y. Xiong. Senguard: Passive user identification on smartphones using multiple sensors. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on*, pages 141–148. IEEE, 2011.

[3] J. Zhu, P. Wu, X. Wang, and J. Zhang. Sensec: Mobile security through passive sensing. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 1128–1133. IEEE, 2013.