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

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 helps protect access to a mobile device from a malicious party by detecting the anomalies early. Our experimental results show that this scheme can distinguish a user-device pair from another with a confidence interval exceeding 80%. The false positive and false negative rates are below 12%. The execution time required for this authentication if of the order of a few milliseconds, which suits mobile devices. Increased data set sizes can push this accuracy into 90+%.

1. INTRODUCTION

Mobile devices are increasingly becoming a repository of all our personal data and credentials. This suggests increased attention to mobile device security. The main gateway to any security schema is user authentication. Biometrics based authentication schemes are less onerous and more transparent than the traditional password based authentication methods. We [15] build a physical unclonable function (PUF) that composes human biometric with silicon biometric leading to a unique user-device pair identity that is robust. This PUF is based on the mobile device touch screen interactions. The challenge is a polyline drawn on the screen. The user traces this challenge line. The human pressure exerted in the trace and the exact traced path profile captures the human biometrics of the user. This pressure is processed through the capacitive touch and sensor circuitry of the touch screen whose output captures the silicon biometrics, in a manner similar to the traditional PUFs. The touch events generated by the Android framework contain pressure values which reflect both the human and the touch screen biometrics. These pressure sequences can be quantized into a binary responses creating a challenge-response authentication framework. This PUF derives its randomness physically - from human behavior and silicon transistor characteristics. The composition of the human and silicon components is not mathematically modelable. This is what makes such an authentication framework robust. This polyline tracing authentication is relatively easy for a user - no passwords to remember and it is fairly transparent.

Mobile device theft - particularly smartphones is a major problem. Consumer Reports [13] reported over 3.1 million smartphone thefts in 2013. Federal Communications Commission (FCC) [4] 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. In NYC and San Francisco, the percentage of crime involving mobile phones shoots upto 55-59%. A mobile device has a time window right after the theft wherein the authentication state still holds, which can be exploited by an adversary for considerable loss. This leads to the need for going beyond discrete time authentication - such as password and touch screen PUF challenge response.

In this paper, we develop a continuous authentication framework based on touch screen based PUFs. Touch screen interactions are an integral part of a mobile device UI. Many mobile apps use a soft keyboard. If these user touch screen interactions can be captured to define a user model on a continuous basis, the user-device pair can be authenticated on a continuous basis. The classical principles of spatial and temporal locality from computer architecture are likely to hold in human behavior. Specific touch screen pressure token sequences representing some phrases from email or messaging apps such as "I am OK" repeat over time giving rise to temporal locality. Spatial locality weakly captures specific word sequences like "the" leading to pressure tokens of "t", "h", and "e". This locality comes from the language constructs for English.

We modified a soft keyboard app to collect all the touch screen keyboard interactions data. Android framework generates a sequence of Motion Event objects in response to these touches. The Android framework includes a class `Mo-`

`tionEvent` (http://developer.android.com/reference/android/view/MotionEvent.html). The `getPressure()` method returns a normalized pressure value in the range $[0, 1]$ which is derived from the quantification of the current flow change due to capacitive change. This pressure value should reflect per device variability in the pressure measurement. The continuous authentication layer (CAL) collects a sequence $S_{initial}$ of N touch events' pressure values $p_0, p_1, \ldots, p_{N-1}$ from the user interaction within an app through the modified keyboard. This sequence is processed for an nth order Markov model $M_{U,D}$ for the given user and device. Figure 4 shows an example of 2-Markov model.

The continuous authentication signature will fail if either the biometrically correct human user or the biometrically correct device component is removed. This paper builds the continuous authentication framework on the user touch screen interactions.

Continuous authentication frameworks' basic 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.

In our system, we record the touch pressures generated by the user through system's soft keyboard. Once we have collected a sufficient number of touches, which is the training phase, we build a user profile or model. Future touch screen tokens are authenticated against this model. 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 through a soft keyboard at a rate of 20-60 words per minute. A word consists of on average five letters leading to a rate of 30*5 touch interactions per minute. This means in the average use case, a user can generate enough data to train the model within 40 minutes of continuous use. Hence the earliest this authentication can kick in for a user-device pair is of the order of 40 minutes.

The structure of the paper is as follows. Section 2 discusses related work. The n-Markov model and its parametrization are discussed in Section 3. Data collection methods are discussed in Section 5. The authentication scheme is presented in Section 4. The results including final numbers and their interpretation are given in Section 6. Conclusions are presented in Section 8.

2. RELATED WORK

There is significant amount of work on PUFs [5], [10], [8] Ratha et al. [12] proposed use of sensor biometrics for authentication In the continuation authentication world, a composite sensor vector [18] has been used to establish a user identity. Sensec builds a user profile from accelerometers, gyroscopes and magnetometers. tOther user biometrics such as inter key stroke timing [17], accelerometer based signature [9], MEMS sensors [1], Gait sensors [3], gestures [7], [14], and broader sensor set [6] have also been used. SenGuard [16] is a similar to SenSec in creating a passive authentication mechanism using sensor based models. In this paper, the continuous authentication is based on a combined user-device identity, which is derived from a user-device (UD)-PUF [15].

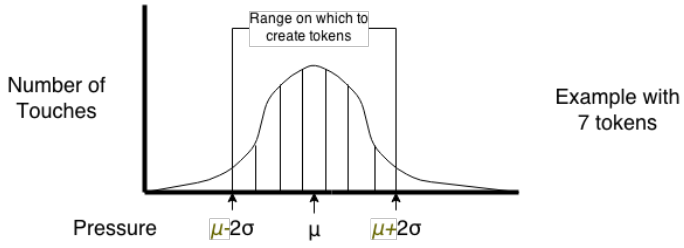

Figure 1: Touch Tokens are Only Created for Pressure Values within Two Sigma Range for Each Key.

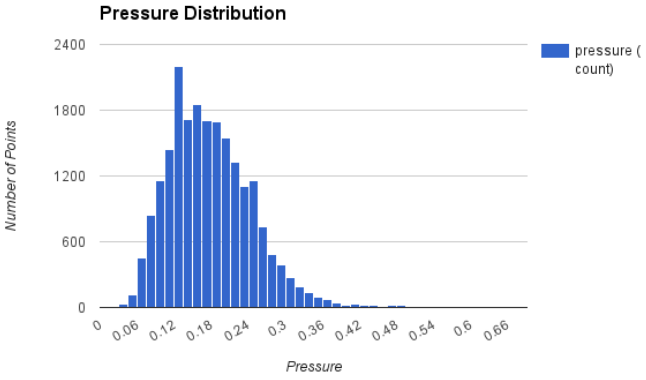

Figure 2: This chart displays a set of pressure data from one of test users. Note that the pressure values are normally distributed.

3. MODELING A USER-DEVICE PAIR

Raw touch interactions at a specific key in the soft keyboard square button generates a raw pressure value in the range $[0, 1]$. These pressure values are tokenized into an input alphabet for the Markov model.

Markov Chains are useful in predicting and modeling systems whose behavior evolves through discrete states. The probability of transitions between states can be captured. In general an n-Markov model states that the user tokens of length n repeat themselves with certain frequency. Hence a user history of n-token sequences could help us classify the user's future behavior. The tokens are touch screen pressure values that are continually generated as the user interacts with an app through a soft keyboard.

Instead of building a complete Markov model of the user behavior, we build a collection of n-grams which are frequently instantiated sequences of n tokens. We use n-sized sequences within a user interaction generated token sequence as an n-gram which approximates n-Markov model. n-grams were originally used in natural language processing [2].

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 pressure values is calculated for each

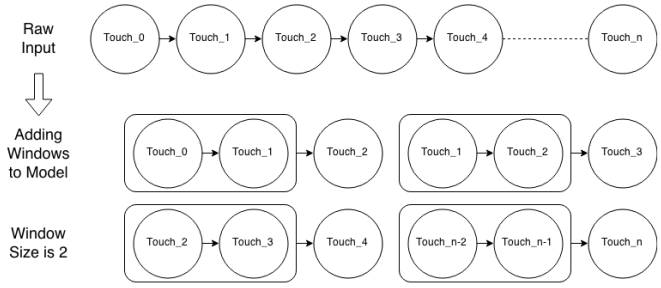

Figure 3: The top of this figure depicts the raw input touchscreen token sequence. Each touch represents a single interaction between the human user and the soft keyboard. The bottom part shows how the raw input is parsed into 2-Markov model. For example, the bottom left image can be interpreted to say that Touch_4 succeeds the sequence Touch_2, Touch_3 with a non-zero probability.

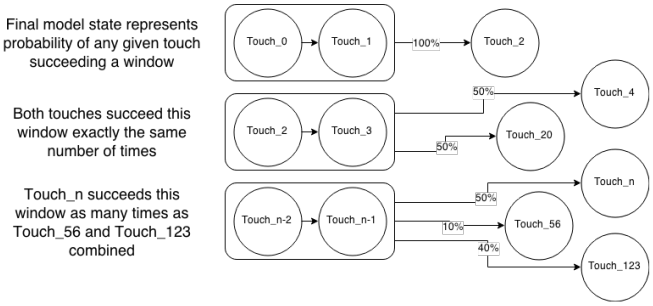

Figure 4: Example 2-Markov models after the probabilities have been calculated. The top model states that Touch_2 succeeds the sequence Touch_0, Touch_1 with probability 100%.

square button area on the touchscreen. In our system these areas correspond to keys of the soft keyboard. In order to develop PUF reproducibility, the pressure tokens from the same key are assumed to have a normal distribution. From the token sequence, a normal distribution mean and variance (μ and σ) values are estimated for each soft keyboard key. If a token pressure value falls outside of $\mu \pm 2 * \sigma$ for a given key, then it is not included in any of the n-token sequences. Figure 1 illustrates this tokenization process. The pressure values falling within $\mu \pm 2 * \sigma$ are tokenized into a predetermined k tokens - 7 token ranges in Figure 1. The value of $\mu \pm 2\sigma$ was chosen because statistically 95.45% of touches will fall withing this range for normally distributed data [11]. Figure 2 plots a set of touch pressure values from one of the test users which shows that this data is close to a normal distribution. Tokens are compared both for the key (keyboard button) location and the tokenized pressure value.

Our goal in modeling user touch screen interactions with a Markov model is to classify the system in terms of its transitions between states. Each touch token contains information about the pressure and location of the interaction with the touchscreen. These token fire off transitions.

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 $count(T, T_i, T_{i+1}, \ldots, T_{i+n-1}) = \sum_{j=0}^{N-n} (1 \text{ if } T_{j..(j+n-1)} == T_{i..(i+n-1)} \&\& T_{j+n} == T)$.

Larger n increases accuracy of the true probability of a given token T following a sequence $T_{i..(i+n-1)}$. The increased accuracy is due to the intuition that two users will be behave identically for a sequence of n tokens. Figure 3 demonstrates how the token sequences for the Markov model are generated from the user's raw input from the soft keyboard.

Recall that the probability of a token T following a token sequence $T_{i..(i+n-1)}$ is expressed as the number of occurrences of T succeeding the given sequence $T_{i..(i+n-1)}$ among all the $N - n + 1$ in-fixes of the sequence $T_{i..(i+n-1)}$. The idea behind this probability calculation is illustrated in Figure 4. Notably, 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 algorithm to calculate the n-gram probabilities of a token T succeeding a given sequenc $T_{i..(i+n-1)}$ is the number of times that touch succeeds the sequence. This counts the number of occurrences of $T_{i..(i+n-1)}||T$, where $||$ denotes concatenation. The use of a prefix tree as a data structure helps keep track of appropriate in-fixes and the affect of backtracking thus increasing the efficiency of this probability calculation.

In our system the n-token sequences are stored in a list while a prefix tree is used to store pointers to the instances of various tree prefixes in the list. The prefix tree nodes also include the count of number of occurrences of a sequence in addition to a list of indexes/pointers to where the sequence occurs in the list storing all the sequences. This index list is useful because it eliminates the need to search the list in order to determine what the successor tokens of a token window are.

4. DIFFERENTIATING USER-DEVICE PAIRS

In distinguishing a user from another user, the n-grams for the two users need to be compared. For a given n-gram $[(T_{i_0} T_{i_1} \ldots T_{i_{n-1}}), (p_0, p_1, \ldots, p_k)]$ where $(T_{i_0} T_{i_1} \ldots T_{i_{n-1}})$ is the prefix sequence and $(p_0, p_1, \ldots, p_k)$ is the probability vector for the next token being Token 0, Token 1, ..., Token k from the alphabet Σ. The distance between two n-grams $distance[(T_{i_0} T_{i_1} \ldots T_{i_{n-1}}), (p_0, p_1, \ldots, p_k)]$ and $[(T_{i_0} T_{i_1} \ldots T_{i_{n-1}}), (q_0, q_1, \ldots, q_k)]$ is given by $\Sigma_{j=0}^{k} |(p_j - q_j)| / (k+1)$ where $|x|$ is the absolute value of x. The difference between two user profiles is the average difference between n-grams belonging

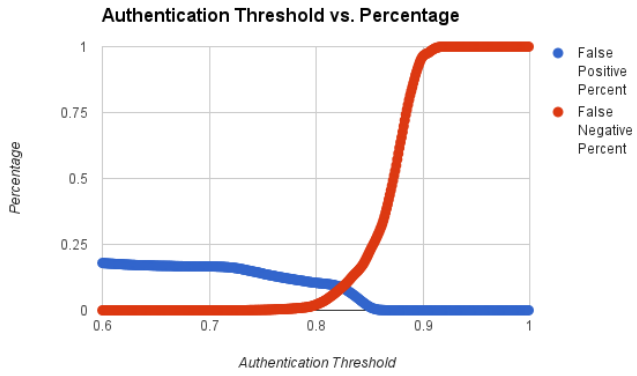

Figure 5: False positive and false negative percentages vary as the authentication threshold is adjusted.

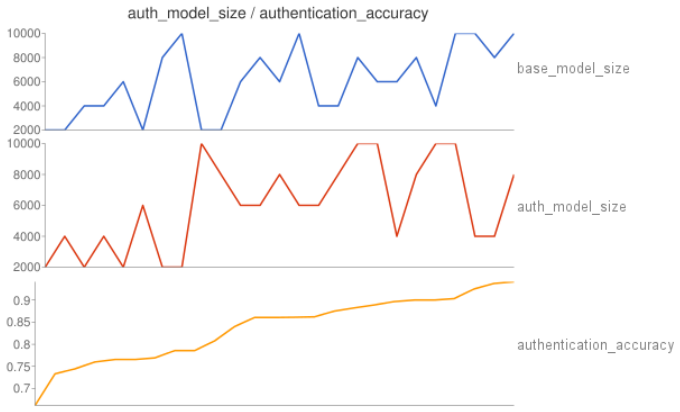

Figure 6: Authentication accuracy is a function of both the base model size and authentication model size.

to the two profiles. If the two n-grams are identical, the distance is 0. One minus this distance is a measure of the *divergence* between two user profiles.

Continuous authentication system needs to determine when two sets of touch pressure values came from the same user-device pair. When authenticating a user, we take the difference between the user profile n-grams constructed from the training data set and the n-grams constructed from the current touch interaction data. The confidence interval that the current user is the same whose training profile exists can be given $1 - avgDistance(TrainingProfile\ n-grams,\ CurrentProfile\ n-grams)$. If the two profiles are identical with distance 0, confidence level in the user identity is highest at 1. If authentication is a binary decision of yes or no, a threshold value $0 \leq th \leq 1$ is set. if the ConfidenceInterval $= 1-$ ProfileDistance $> th$, the user is authenticated; otherwise not. Figure 5 illustrates how false positive percentage and false negative percentage rates vary based on the threshold value for authentication.

False positive rate measures the fraction of authentications between two touch token profiles where the two profiles did not come from the same user-device pair., False negative rate is exactly the inverse of false positive rate in that it describes the frequency with which the two touch token profiles from the same (user, device) pair fails the authentication.

In Figure 5 there exists a clear intersection between false negative and false positive percentages. This intersection is significant; at this point the system is not biased towards either of false positive or false negative events. This point represents a balance in design. For our data sets, this balance point was approximately at $th = 0.83$. For a threshold in the range $0.6 \leq th \leq 0.83$, false negative rate was close to 0 at the cost of a false positive rate approaching 0.2. The choice of this threshold will depend on the authentication requirements.

5. DATA COLLECTION AND ANALYSIS

Data for 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 normally would. 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 to train the profile.

The results presented here were derived from the touch data generated by two users. These users each used two different devices creating four user-device pairs each having a large number of touches. The size of the data sets collected were at least 4500 and at most 47500.

6. RESULTS

The two key parameters left to determine are the training data set size (base model) and the authentication data set size (authentication model). The figure of merit is authentication accuracy. Figure 6 shows *authentication_accuracy* as a function of *base_model_size* and *auth_model_size*. We define *authentication_accuracy* to be the percentage of authentications for which our system makes the correct decision (both false positives and false negatives count against the accuracy). 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.

Both the base model and the authentication model sizes were varied between 2000-10,000 touches. In some instances, increased numbers of touches did not result in higher authentication accuracies. Part of this has to do with the way the distance between the base model and the authentication model. Existence of an n-gram in authentication model that does not belong to the base model is considered an unlikely event, and hence an anomaly. It is penalized in distance computation by computing the distance as its probability $Prob(NG[i, auth])$ - probability of an n-gram i in the authentication model. Hence if this probability is .75, its contribution to confidence interval is only $1 - 0.75 = 0.25$. An authentication model whose size exceeds the base model size by a significant margin usually does not benefit the authentication accuracy. The authentication model includes some

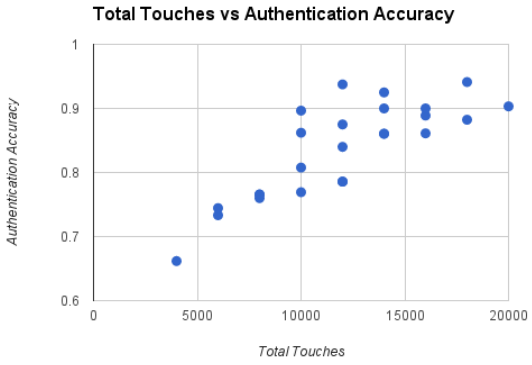

Figure 7: Authentication accuracy is linked more closely to the total number of touches used in the authentication then to either the base model size or the authentication model size. Although authentication accuracy does not strictly increase with the number of touches it does trend upward.

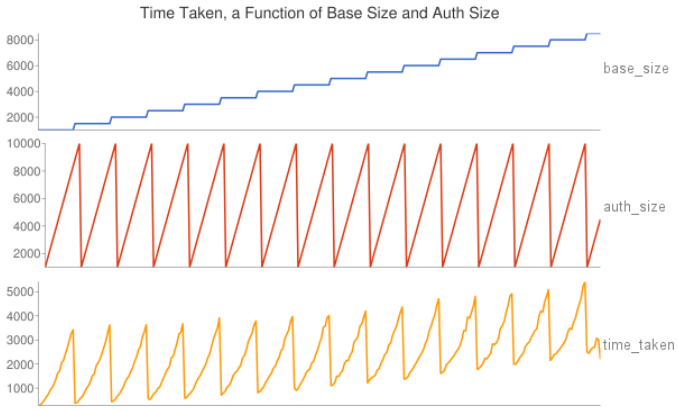

Figure 8: The authentication time on a Nexus 7 tablet as a function of base model size and auth model size. Time taken is measured in milliseconds while model sizes are measured by number of touches.

n-gram sequences that were not seen in the training base model. Hence, they do not add much to the authentication accuracy.

A small sized base model suffers in the authentication accuracy for the same reasons. The number of n-grams is low in the base model. Then the difference value between the base model and the authentication model will be high due to the penalty assessed on the authentication model n-grams that do not exist in the base model.

Figure 7 presents the tradeoff between the amount of data needed for an authentication and the accuracy of that authentication. In general, more data will yield a better authentication accuracy, but this is not always true. The size of the base model seems to contribute more to authentication accuracy then does the size of the authentication model, therefore, if the goal is to increase authentication accuracy it is better to increase the size of the base model compared to increasing the size of the authentication model.

Figure 8 displays the execution time of our system on a Nexus 7 tablet. Time taken is measured in milliseconds while model sizes are measured by number of touches used to construct the model. The time 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 computationally intensive activity - more of a UI event. Time does include the probability computation for each of the models and the comparison between the models. This chart is a good representation of how each of the model sizes affects the execution time of the system. The overall time taken is trending upward linearly in the total base model size and auth model size. This suggests that the total number of touches used, that is the sum of base model size and auth model size, dictates the authorization efficiency. The time dependence on the total number of touches makes sense because the probability computations in Markov model dominate. Also of note, the base model size and auth model size

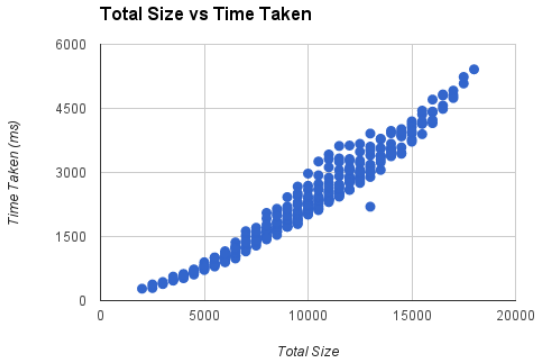

Figure 9: The execution time on a Nexus 7 tablet as a function of base model size plus auth model size. Time taken is measured in milliseconds while model sizes are measured by number of touches.

seem to affect time taken differently. A change of factor 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 execution time depends on the total number of touches used in creating the models. The trend suggests that the total time increases nonlinearly as the number of touches used to generate the model increases. This figure also supports the conclusion that the total size of the model in number of touches has a larger influence on the execution time then the sizes of either the base or auth models individually.

Figures 6 & 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 of highest authentication accuracy corresponding to the largest numbers of touches. Figures 8 and 9 demonstrate the performance tradeoff associated with 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 time.

7. DISTRIBUTED USER-DEVICE (UD-) PUF

The preceding discussion extends the human-device entangled PUFs [15] to a distributed implementation. The n-Markov model also determines frequently occurring n-token sequences for a given user. All of these n-token sequences can be considered to be the challenge set. The resulting pressure value responses can then be quantized into a binary response in the same way as in [15] resulting in the same variability and reproducibility properties. The main difference is that this distributed PUF is derived as a side-effect of the authentication mechanism. This could support alternate functionalities that can benefit from a PUF based on the user-device physical randomness.

8. CONCLUSIONS

This paper presents a continuous authentication approach which utilizes the variability in the way users interact with the touchscreens of their devices to differentiate distinct user-device combinations. A continuous authentication model prevents data theft from a mobile device even for lost devices.

our experiments establish optimal training data set sizes around 6000-10000 touches, an authentication data set size at around 2000 touches, and the n-gram sequence length in the range 3-6. This leads to authentication accuracy over 80% with both false positive and false negative rates contained below 12%.

9. REFERENCES

[1] A. Aysu, N. F. Ghalaty, Z. Franklin, M. P. Yali, and P. Schaumont. Digital fingerprints for low-cost platforms using mems sensors. In *Proceedings of the Workshop on Embedded Systems Security*, WESS '13, pages 2:1–2:6, New York, NY, USA, 2013. ACM.

[2] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, Dec. 1992.

[3] S. Choi, I.-H. Youn, R. LeMay, S. Burns, and J.-H. Youn. Biometric gait recognition based on wireless acceleration sensor using k-nearest neighbor classification. In *Computing, Networking and Communications (ICNC), 2014 International Conference on*, pages 1091–1095, Feb 2014.

[4] F. C. 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.

[5] S. Devadas. Physical unclonable functions and secure processors. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '09, pages 65–65, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] S. Dey, N. Roy, W. Xu, and S. Nelakuditi. Acm hotmobile 2013 poster: Leveraging imperfections of sensors for fingerprinting smartphones. *SIGMOBILE Mob. Comput. Commun. Rev.*, 17(3):21–22, Nov. 2013.

[7] T. Feng, Z. Liu, K.-A. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. Nguyen. Continuous mobile authentication using touchscreen gestures. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pages 451–456, Nov 2012.

[8] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 148–160, New York, NY, USA, 2002. ACM.

[9] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive Mob. Comput.*, 5(6):657–675, Dec. 2009.

[10] D. Merli, G. Sigl, and C. Eckert. Identities for embedded systems enabled by physical unclonable functions. In M. Fischlin and S. Katzenbeisser, editors, *Number Theory and Cryptography*, volume 8260 of *Lecture Notes in Computer Science*, pages 125–138. Springer Berlin Heidelberg, 2013.

[11] F. Pukelsheim. The Three Sigma Rule. *The American Statistician*, 48(2):88–91, May 1994.

[12] N. K. Ratha, J. H. Connell, and R. M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM Syst. J.*, 40(3):614–634, Mar. 2001.

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

[14] N. Sae-Bae, N. Memon, K. Isbister, and K. Ahmed. Multitouch gesture-based authentication. *Information Forensics and Security, IEEE Transactions on*, 9(4):568–582, April 2014.

[15] 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.

[16] 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.

[17] Z. Syed, S. Banerjee, and B. Cukic. Leveraging variations in event sequences in keystroke-dynamics authentication systems. *Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05)*, 0:9–16, 2014.

[18] 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.