Toward Robotic Robbery on the Touch Screen

ABDUL SERWADDA, Texas Tech University
VIR V. PHOHA, Syracuse University
ZIBO WANG, Louisiana Tech University
RAJESH KUMAR and DIKSHA SHUKLA, Syracuse University

14

Despite the tremendous amount of research fronting the use of touch gestures as a mechanism of continuous authentication on smart phones, very little research has been conducted to evaluate how these systems could behave if attacked by sophisticated adversaries. In this article, we present two Lego-driven robotic attacks on touch-based authentication: a population statistics–driven attack and a user-tailored attack. The population statistics–driven attack is based on patterns gleaned from a large population of users, whereas the user-tailored attack is launched based on samples stolen from the victim. Both attacks are launched by a Lego robot that is trained on how to swipe on the touch screen. Using seven verification algorithms and a large dataset of users, we show that the attacks cause the system's mean false acceptance rate (FAR) to increase by up to fivefold relative to the mean FAR seen under the standard zero-effort impostor attack. The article demonstrates the threat that robots pose to touch-based authentication and provides compelling evidence as to why the zero-effort attack should cease to be used as the benchmark for touch-based authentication systems.

Categories and Subject Descriptors: C.2 [**Computer-Communication Networks**]: Security and Protection

General Terms: Security

Additional Key Words and Phrases: Touch gestures, behavioral biometrics, robotic attacks, smartphone security

ACM Reference Format:
Abdul Serwadda, Vir V. Phoha, Zibo Wang, Rajesh Kumar, and Diksha Shukla. 2016. Toward robotic robbery on the touch screen. ACM Trans. Inf. Syst. Secur. 18, 4, Article 14 (May 2016), 25 pages.
DOI: http://dx.doi.org/10.1145/2898353

1. INTRODUCTION

The zero-effort attack—a form of impostor attack in which a given user's samples are used to test another user's biometric template (without any forgery whatsoever)—is the defacto performance evaluation method for touch-based authentication systems on smart phones and other touch input devices (e.g., see Govindarajan et al. [2013], Alexander et al. [2012], Feng et al. [2012], and Frank et al. [2013]). In this article, we argue that this kind of attack majorly underestimates the magnitude of the threat that a touch-based authentication system could face in practice. In particular, we show that a robot costing less than $400 on the open market can be trained to swipe (i.e.,

This work was supported in part by DARPA Active Authentication contract: FA 8750-13-2-0274 and by National Science Foundation Award Number: 1527795.
Authors' addresses: A. Serwadda, 211H, Computer Science Department, Texas Tech University, TX 79414; email: abdul.serwadda@ttu.edu; V. V. Phoha, CST 4-206, EE & CS, Syracuse University, NY 13210; email: vvphoha@syr.edu; Z. Wang, Department of Engineering and Science, Louisiana Tech University, 305 Wisteria Street, P.O. Box 3178, Ruston, LA 71272; email: zwa006@latech.edu; R. Kumar and D. Shukla, 4-178, Center for Science and Technology, Syracuse University, NY 13210; emails: {rkuma102, dshukla}@syr.edu.

DOI: http://dx.doi.org/10.1145/2898353

originate touch gestures) in such a way as to cause a touch-based authentication system to perform much worse than depicted by the traditional zero-effort attack.

We showcase two types of Lego-driven robotic attacks: (1) population statistics–based attack, in which patterns gleaned from a large database are used to formulate the input supplied to the robot, and (2) user-tailored attack, which uses samples stolen from the victim in question as a basis to formulate input to the robot. Using seven state-of-the-art classification algorithms, we rigorously analyze the impact of the attacks, showing that both attacks significantly degrade the performance of a touch-based authentication system.

Perhaps the major indicator of the feasibility of our attacks is the ease with which they could be launched. The Lego robot used to implement the attack can be easily operated by a middle school student (see Mauch [2001]), whereas the population data required to launch the population-based version of the attack is easily accessible on the Web. Further, no knowledge of the features and/or underlying software implementation is needed, as the robot just has to swipe on the phone screen with no other technicalities required to break the authentication system. This extent of simplicity of the attack is in stark contrast to common biometric spoofing attacks whose design typically requires a deep understanding of the theory and techniques used in the domain (e.g., see Uludag and Jain [2004]) in addition to intricate software manipulations in some cases (e.g., see Khandaker et al. [2013]).

The following are the contributions of this article:

(1) We propose a new family of attacks for the performance evaluation of touch-based authentication systems. Based on the Lego robot, we present both an algorithmic and mechanical design for the attack. Given the success rates of the attack and the ease with which it could be launched by an adversary, our belief is that this kind of attack will become the standard impostor testing method for touch-based authentication. Our designs should thus serve as a blueprint to refinements of this attack that researchers will apply to different touch-based authentication systems or different robots.
(2) We thoroughly explore the impact of the attack on the average classification accuracy seen across the population and each user's performance. We show that the population-based attack increases the mean false acceptance rate (FAR) of the majority of the classification algorithms by more than 100% of the mean FAR seen before the attack. The user-specific attack even has a much more drastic effect on the authentication system, driving the postattack classifier FAR to almost 1 (with FAR measured on the scale 0 to 1) in some of the cases studied in the article.
(3) We perform a rigorous analysis of the discriminability properties of the features used for our classification system. Because touch-based authentication does not yet have a standard set of widely used features, our feature analysis will help to improve the community's understanding of the properties of the different features that a touch-based authentication system could be designed to use.

The rest of the article is organized as follows. We discuss related work in Section 2, our data collection experiments and feature analysis in Section 3, and the design of the robotic attacks in Section 4. We finally present our performance evaluation framework in Section 5, the results of the attacks in Section 6, and our conclusions in Section 7.

2. RELATED WORK

This article builds upon a previous conference paper (see Serwadda and Phoha [2013b]), in which we first showcased a Lego robot breaching a touch-based authentication system. The current work covers a much wider scope than our previous work. The important differences between this work and the previous paper are as follows:

(1) We introduce the Lego-driven user-tailored robotic attack and compare it to the generic population statistics–driven attack that we studied in the previous work. The former attack uses input derived from swiping samples stolen from the victim in question, whereas the latter uses inputs gleaned from the general population. Comparison of the two attacks gives interesting new insights into the relative impact of the different kinds of threats facing touch-based authentication systems.
(2) We analyze the impact of both user-specific attacks and population statistics–based attacks on seven different classification algorithms. In the previous work, our performance evaluation was based on only two classification algorithms. Our evaluation of the attack based on a wide range of classifiers helps to address the question of whether certain classifiers, by virtue of differences in operational philosophies, could be particularly vulnerable or resistant to the attack.
(3) We perform a thorough analysis of the discriminative power of different touch biometrics features. Findings from the feature analysis help to provide a better understanding of the quality of different touch biometrics features. In the previous work, we defined a set of features and subjected them to the robotic attack without any feature analysis or feature selection whatsoever.
(4) We redesign the robot to further emphasize the simplicity of the attack. In the conference version, we implemented the attack algorithm using BricxCC, an IDE that provides a programming language similar to the C language. In this work, we present a design that can very easily be implemented based on the drag and drop Lego IDE used for instruction to novice programmers (e.g., middle school students [Mauch 2001]). The fact that the design presented in this work can easily be implemented by a novice programmer helps to cement the argument that this family of attacks represents a major threat to touch-based authentication.

We categorize the rest of the related works into two families: those studying the performance of touch-based authentication systems and those studying algorithmic attacks on behavioral biometric authentication.

2.1. Touch-Based Authentication

There is now a fast-growing body of work on touch-based authentication. This body of work is categorized into two parts: (1) single-point authentication, a form of authentication in which touch gestures are only monitored at the entry point to an application (e.g., at login), and (2) continuous authentication, a form of authentication in which touch gestures are monitored throughout a user's session on a touchscreen device. Here, we only survey past works on continuous authentication, as this is the type of authentication for which our attack is designed and evaluated.

Frank et al. [2013] obtained median equal error rates (EERs) of between 0% and 4% when they used a k-nearest neighbor (kNN) classifier and a support vector machine (SVM) to classify touch strokes generated by a group of 41 users. The lowest EERs, of about 0%, were obtained when both training and testing data were drawn from a single phone usage session, whereas the highest EERs, of approximately 4%, were obtained when training and testing data were collected 1 week apart. When training and testing data were drawn from two sessions on the same day, EERs were between 2% and 3%.

In Serwadda et al. [2013], we compared the performance of 10 classification algorithms for touch-based authentication. The comparison was done under the zero-effort attack model (i.e., without the use of robotic forgeries). The lowest EER of 10.5% was obtained with the logistic regression classifier, whereas the highest EER of 42% was obtained with the J48 tree. With the aid of a series of statistical tests of significance, it was found that the random forest and SVM verifiers were comparable to the logistic regression verifier in overall performance. Feng et al. [2012] used a dataset of 40 users

and obtained an FAR of approximately 7.5% at a false rejection rate (FRR) of approximately 8% for the random forest verifier, which performed best among a set of three verifiers. When users wore a digital glove, which provided more information about the movement of users' hands, the FARs and FRRs of all classifiers in the study were found to drastically reduce.

More recently, Zhao et al. [2013] proposed a new graphic feature that expresses touch strokes in terms of shapes and intensity values that are used to create an image. The feature was found to be highly discriminative, as an EER of about 2% was obtained when swiping and zooming features were fused.

All works cited previously were based on biometrics data collected in a controlled setup where users interacted with specifically designed applications. The work in Li et al. [2013] differs from this common experimental design by virtue of studying the performance of a live touch-based authentication system in which users took the phones with them and freely interacted with any applications of their choice. To be able to monitor the touch gestures associated with all applications installed on the phone, the authors exploited a "hack" in the lower layers of the Android system. The authentication system, evaluated based on a group of 75 users and an SVM classifier, was found to attain up to 75% classification accuracy.

The classification performance reported in all five works cited earlier assumes a naive adversary who makes no attempt to forge a given swiping pattern. Our work advances the state of the art by proposing a more rigorous family of attacks that more closely simulate the kinds of adversarial technologies being used against computing systems today.

2.2. Algorithmic Attacks on Behavioral Authentication Systems

Several recently proposed attacks on behavioral biometrics systems share the same motivation as our work. Ballard et al. [2007] used population statistics as input to an algorithmic attack on a handwriting biometrics system. Relative to the attacks launched by trained forgers, the attack was found to be much more effective for certain users. In Serwadda and Phoha [2013a], an algorithmic attack that leverages generic information extracted from a large database of typing data was shown to severely degrade the performance of password-based (or short text) keystroke authentication system. Compared to the standard zero-effort impostor attack, the algorithmic attack was shown to cause a greater than 80% increment in the mean EER when the keystroke profiles being attacked were built from a short seven-character string.

Meng et al. [2013], Wang et al. [2012], and Khandaker et al. [2013] propose three attacks on keystroke authentication. All three attacks assume an adversary who steals some typing samples from the intended victim before using them as input to the attack algorithm. Meng et al. [2013] used the stolen samples as a basis for training humans who iteratively forge the intended victim's pattern, whereas Wang et al. [2012] used the stolen samples to launch attacks that distort the victim's template and expose it to further attacks. Khandaker et al. [2013], on the other hand, simply replayed the stolen samples into the authentication system. In the most extreme case, the attacks in Meng et al. [2013], Wang et al. [2012], and Khandaker et al. [2013] respectively caused the system EER to increase by 395%, 305%, and 2,730.55% relative to the zero-effort attack.

Just like our work, these attacks provide evidence that it is no longer adequate to evaluate behavioral biometrics systems under the assumption of a naive adversary. Notably, however, our attack is clearly different from these attacks, as it occurs in the analog domain, requires very little expertise on the part of the attacker, and does not require extraordinary privileges, such as attacks in Khandaker et al. [2013] where the adversary is assumed to successfully install a keylogger on the victim's computer.

3. DATA COLLECTION, FEATURE EXTRACTION, AND FEATURE ANALYSIS

3.1. Data Collection and Feature Extraction

Following IRB approval (IRB HUC-1086 from Louisiana Tech University), we collected data using an Android app that recorded users' swiping patterns while they answered a set of multiple choice questions. Although the target activity in this experiment was the answering of multiple choice questions, we designed the experiment to simulate a broad range of applications that involve users processing content on the screen while they browse (e.g., email, Web browsing). In particular the multiple choice questions were organized in such a way that users had to carefully browse back and forth while reading and processing text read on the phone screen so as to locate the segment of text that contained the solution (or clues to the solution). To ensure that users did not just aimlessly browse around and select random answers to the questions, the app computed a score that expressed the number of correct answers as a fraction of the total number of questions in the exercise. We found that the expectation to receive a grade at the end of the exercise motivated users to focus on the exercise and carefully undertake all tasks assigned.

The majority of participants were students between 18 and 25 years of age. All data was collected using Google Nexus S phones. Participants were notified about the study through mass emails that were sent out to the whole university. As part of our IRB requirements, the users were told about the purpose of the experiments in advance. Each user participated in this exercise over two sessions that were one or more days apart. Our decision to have each user provide data over multiple days (as opposed to a single session on a single day) was to capture some of the variability in behavior that a user might have from one day to the next. Data from one session was used for classifier training, whereas data from the other was used for classifier testing (more details on the training and testing process are discussed in Section 5.2). To eliminate the possibility of users memorizing the solutions and answering questions without having to swipe back and forth to find the solutions, users had to answer a different set of questions during each session.

While each user swiped, the app recorded at regular intervals the coordinates of the point that the finger touched, the pressure exerted by the finger on the screen, the area between the finger and the screen, the orientation of the phone (i.e., portrait or landscape), and the time at which the finger touched the point in question. From these raw readings, we extracted a wide range of features that draw from those used in past works on touch-based authentication (e.g., see Li et al. [2013], Serwadda et al. [2013], Serwadda and Phoha [2013b], and Frank et al. [2013]). As done in past work (e.g., see Frank et al. [2013]), feature computation was done after the elimination of very short strokes (comprising three or fewer touch points), as they were likely a result of clicking (or tapping) as opposed to swiping. A description of the extracted features follows.

Over a complete touch stroke, a vector corresponding to each of the previously described raw measurements (i.e., coordinates, area, pressure) was fully populated. From each of the area and pressure vectors, we computed 5 features, namely the first, second, and third quartiles and the mean and standard deviation of the elements in the vector (i.e., a total of 10 features). Using the time and coordinate vectors, we computed a velocity vector, which we in turn used to compute an acceleration vector. From the velocity and acceleration vectors, we again computed the earlier mentioned 5 features, creating another 10 features. Finally, for each stroke, we computed the following 8 features to make a total of 28 features: the x-coordinate of the start-point, the x-coordinate of the end-point, the y-coordinate of the start-point, the y-coordinate of the end-point, the time taken to complete a stroke, the direct distance between the ends of a stroke, the sum of the distances between every pair of adjacent points on a

Table I. IDs Assigned to the Features Used to Represent a Touch Stroke

ID	Feature Description
1	x-coordinate of the start-point
2	y-coordinate of the start-point
3	x-coordinate of the end-point
4	y-coordinate of the end-point
5	Direct distance between ends of a stroke
6	Time taken to complete a stroke
7	Tangent of the angle between the line joining the end-points and the horizontal
8	Sum of the distances between every pair
9	Mean velocity
10	Velocity standard deviation
11	Velocity quartile #1
12	Velocity quartile #2
13	Velocity quartile #3
14	Mean acceleration
15	Acceleration standard deviation
16	Acceleration quartile #1
17	Acceleration quartile #2
18	Acceleration quartile #3
19	Mean pressure
20	Pressure standard deviation
21	Pressure quartile #1
22	Pressure quartile #2
23	Pressure quartile #3
24	Mean area
25	Area standard deviation
26	Area quartile #1
27	Area quartile #2
28	Area quartile #3

stroke, and the tangent of the angle between the line joining the end-points of a stroke and the horizontal. Table I summarizes the full list of features.

In the following sub-section we analyze the informativeness of each of the 28 features before selecting the final set of features used for the authentication system.

3.2. Feature Analysis and Selection

To gain insights into the properties of the 28 extracted features, we studied three measures of feature quality: the relative mutual information between each feature and the class labels, comparison between feature distributions, and the correlation coefficients between features. Before performing the feature quality investigations, we first categorized each user's touch strokes into four categories: portrait-vertical strokes, portrait-horizontal strokes, landscape-vertical strokes, and landscape-horizontal strokes. The meanings of theses stroke categories are easily inferred from the names. For example, portrait-vertical strokes are those strokes that were executed to move screen content vertically while the phone was held in a portrait orientation. We make distinctions between these stroke types to ensure meaningful feature comparisons during the feature quality measure computations. For example, comparing a feature vector derived from a portrait-horizontal stroke with that derived from a portrait-vertical stroke would be meaningless, as certain features (e.g., the start coordinates) change depending on how the phone is held. In the feature analysis investigations that follow in the following

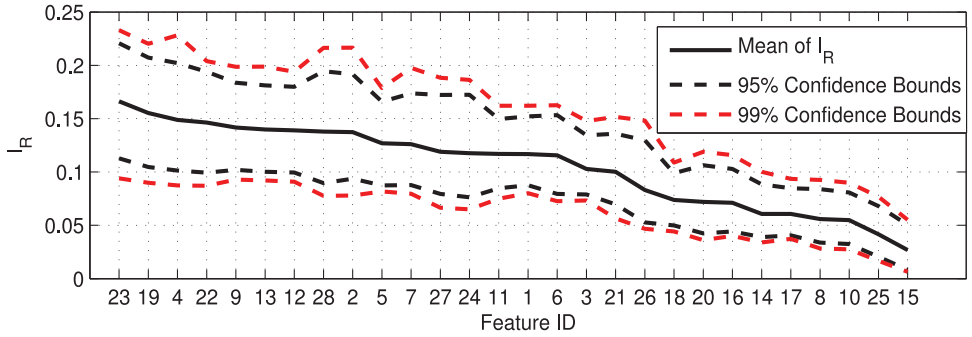


Fig. 1. The 95% and 99% confidence bounds of I_R for the horizontal strokes generated when the phone was held in a portrait orientation. Bootstrap resampling was done 1,000 times, with data from a sample size of 35 randomly selected users being used to compute each bootstrap estimate.

three sections, we only present findings from the horizontal-portrait strokes, as they capture the general trends that we observed in our dataset. We do not discuss the other stroke categories, as they provide no significant new insights.

3.2.1. Relative Mutual Information. Let F denote the random variable representing the outputs of a feature across the population, and let C denote the random variable representing the class labels. The quantity $I_R = \frac{I(F;\, C)}{H(C)}$ is a measure of the relative mutual information between F and C if $I(F;\, C)$ is the mutual information between F and C and $H(C)$ is the entropy of C. I_R assumes values between 0 and 1, with values close to 1 indicating a feature that is very informative. Frank et al. [2013] also used this measure of informativeness to analyze the properties of their features.

However, we take a different perspective of I_R from that studied by Frank et al. [2013]. In particular we use bootstrap resampling to compute the mean of I_R and the 95% and 99% confidence intervals of I_R for each feature. Frank et al. computed a single value of I_R for each of their features. In general, the bootstrap bounds give a more rigorous understanding of the performance limits of the features compared to a single value of I_R computed for each feature across the population. Further, our mean value of I_R computed from random subsamples of the population is likely to represent the performance of a feature in more general terms than a value of I_R computed once for a fixed set of users. We use the standard nonparametric bootstrapping method in Bolle et al. [2000] with a random sample size of 35 users for each of 1,000 repetitions. Because this method does not make any assumptions about the underlying statistical distributions of the data, the bounds reported from this analysis should very closely represent the true behavior of the features. Figure 1 shows the results of our bootstrap analysis for the horizontal strokes in the portrait mode. To compute I_R, we first discretize each feature using 20 equally spaced bins. The figure reveals a clear distinction in the mean value of I_R between the best-performing features (e.g., features 23 (pressure quartile #3), 19 (mean pressure), 4 (y-coordinate of end-point), and 22 (pressure quartile #2)) and the worst-performing features (e.g., features 17 (acceleration quartile #1), 8 (sum of distances between each pair), 10 (velocity standard deviation), 25 (area standard deviation), and 15 (acceleration standard deviation)). The figure also shows that the best-performing features had wider confidence bounds on I_R than the worst-performing features, an indication that the poor-performing features were consistently poor during the various bootstrap runs. This trait enables us to more concretely conclude about the identities of the poor features, which in turn helps inform the decisions to eliminate certain features.

Table II. Selection of the Features That for a Large Number of User Pairs Did Not Provide Enough Evidence to Enable Us to Reject the Hypothesis That the Underlying Feature Distributions Did Not Differ Across Users. In Practice, Such Features are Likely to be Poor at Separating Users

Feature ID	Feature Name	User Pairs for Whom H_o Not Rejected (%)
15	Acceleration standard deviation	40
16	Acceleration quartile #1	33
17	Acceleration quartile #2	28
18	Acceleration quartile #3	31
25	Area standard deviation	30

3.2.2. Underlying Feature Distributions. For a given pair of users, a feature is likely to be more discriminative if the underlying statistical distributions of the users' feature values are distinct from each other. If the distribution is the same for two users, such a feature is likely to be poor at separating the users in question. We conducted an analysis of the statistical distributions exhibited by the features to complement our findings from the mutual information-based analysis.

We used the two-sample Kolmogorov-Smirnov (K-S) test [Gail and Green 1976] for this analysis. Based on 100 randomly selected user pairs, we carried out this test for each of our 28 features. In each test, the null hypothesis was that feature data from the two users being compared had the same underlying continuous distribution. Failure to reject the null hypothesis means that the users have the same distribution for the feature in question. Table II shows a summary of the results for the features that performed worst. For the 100 user pairs, we count the number of user pairs for whom H_o cannot be rejected for each of the four categories of strokes (i.e., portrait-vertical, landscape-vertical, portrait-horizontal, and landscape-horizontal). We tabulate the highest average values across the four stroke categories.

The table reveals that most of the poor-performing features under the mutual information criteria also were featured among the worst performers under the K-S test procedure. For example, the standard deviation of the finger acceleration (feature 15) that had H_o rejected in 40% of the cases also had the lowest value of I_R in Figure 1. Likewise, the three acceleration quartiles (features 16 through 18) and the area standard deviation (feature 25) performed poorly based on the I_R criteria (see Figure 1).

3.2.3. Correlation Coefficients Between Features. Figure 2 represents the correlation coefficients between each pair of features. The color bar on the right maps colors to actual correlation coefficients. The figure shows that a good number of features are very slightly correlated with each other (i.e., correlation coefficients approximately equal to zero). However, there are some very strong correlations seen for feature pairs comprised of features whose IDs are between 19 and 28 (see Table I). These features actually correspond to statistical measures extracted from the pressure and area vectors, for which correlation is not surprising given the theoretical relationship between these measures.

Features selected. For our final feature set, we drop feature 15 (acceleration standard deviation), which performed worst in all of our rankings and the five area metrics due to their strong correlations with the pressure metrics. Although a set of correlated features may perform well if used as part of a larger set of features, we drop these features to speed up learning, as was done in Frank et al. [2013]. Because a continuous authentication system is expected to run all the time, our use of a compact feature set should be in line with the need to minimize the amount of resources that the authentication application could consume in a setting where a good number of potentially resource-intensive applications could be competing for resources.

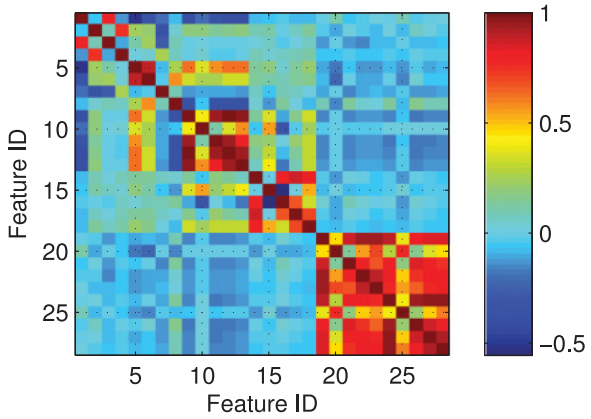


Fig. 2. Correlation plot for the 28 features.

4. ATTACK DESIGN

4.1. General Assumptions

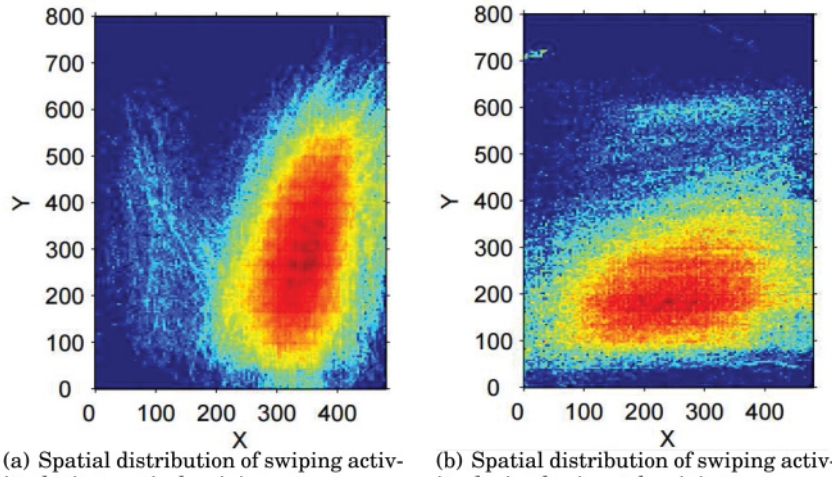
Our attacks assume an adversary who gets physical access to a phone for which touch-based continuous authentication is the only active layer of defense. In practice, this scenario may arise for an attacker who (1) breaks the PIN lock mechanism (e.g., using methods such as those in Aviv et al. [2010] and Owusu et al. [2012]), (2) finds a phone in which the PIN lock has been disabled temporarily (e.g., a user who sets a very long timeout for the PIN lock), or (3) finds a phone in which the PIN lock has been completely disabled by the owner [Frank et al. 2013]. To be able to determine the amount of extra security that touch-based continuous authentication adds to the standard PIN lock in the worst case, we believe that these assumptions must necessarily be made. Ballard et al. [2008] present an investigation in which a similar assumption (i.e., that the adversary has access to the victim's password) was made to enable rigorous evaluation of the security of randomized biometric templates (RBTs).

In the attack itself, the attacker will seek to view private information on the phone (e.g., emails, pictures) without triggering the anomaly detection mechanism. The attack thus basically proceeds by scrolling/swiping through documents on the phone. In practice, we believe that the attacker could even assist the robot during certain operations (e.g., occasionally clicking at a challenging location), as the anomaly detector will most likely not be sensitive enough to detect a few anomalous clicks.

4.2. Examining Users' Swiping Behavior

To design the attacks, we first examine the way in which people swipe in general. How random is swiping behavior across a population? Are there certain distinct traits that manifest frequently across a large number of users? This section provides answers to these and related questions. We present results on the pressure exerted on the screen, the area between the finger and the screen, and the region of the phone at which most swiping is done.

4.2.1. Location of Swiping Activity. Figure 3 shows the density of touch strokes captured at different positions of the phone screen. The dark blue color corresponds to regions that saw very little or no swiping/scrolling activity, whereas a high intensity of red corresponds to regions that saw a lot of swiping. Observe (Figure 3(a)) that most vertical strokes generated by our user population originated from points having x values in the neighborhood of 300 units and terminated at a position with an x value



(a) Spatial distribution of swiping activity during vertical swiping

(b) Spatial distribution of swiping activity during horizontal swiping

Fig. 3. Color map showing the spatial distribution of touch strokes on the phone screen. A high intensity of blue corresponds to regions that saw very few strokes, whereas a high intensity of red implies a region that saw intense swiping activity. The phone was being used in portrait mode when the strokes were generated. Note that the coordinate system used on the figures is different from that used by the Android system.

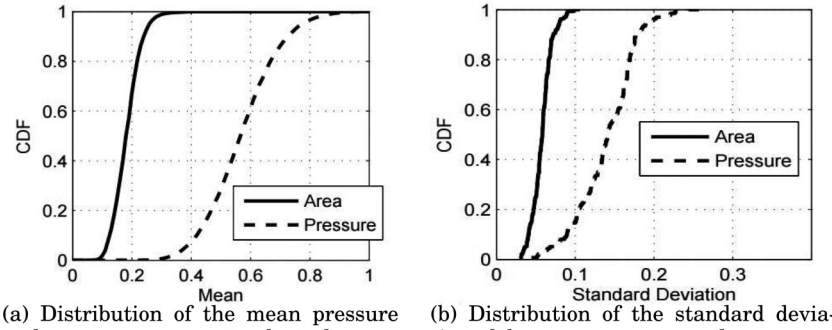
close to 400 units (and vice versa). Notably, this region of high activity comprises less than 50% of the screen display. The heart of the red region (which tends toward black) occupies an even much smaller portion of the screen. Similar traits (see Figure 3(b)) were seen with horizontal swiping.

Based on evidence provided through this plot, an adversary with access to general population statistics could potentially significantly narrow down the scope of features, such as (1) the x-coordinate of the start-point of a stroke, (2) the y-coordinate of the start-point of a stroke, (3) the x-coordinate of the end-point of a stroke, (3) the y-coordinate of the end-point of a stroke, (4) the duration of a stroke, (5) the summation of distances between consecutive points of a stroke, and (6) the direction of the end-to-end line, among other features. These features represent a good proportion of the features used to characterize users' touch gestures in past research (e.g., see Frank et al. [2013] and Li et al. [2013]) and will also be used in this study.

Regarding the cause of the clustering tendency, our conjecture is that the high density of strokes on the right side of the screen (i.e., taking the case of vertical swiping) was likely because the majority of users are right handed, tending to hold the phone in the right hand and swiping with the thumb, or holding the phone in the left hand and swiping using one of the fingers on the right hand.

In any of these two scenarios, a user is very likely to swipe in the manner reflected in the figure. We do not rule out the possibility that certain applications could depict variations from the pattern shown in the figure. In this case, we argue that a committed attacker who has interest in breaking into such applications can undertake research on the swiping traits associated with these applications.

4.2.2. Finger Area and Pressure on the Screen. Figure 4(a) shows the distribution of the mean area touched by the finger and the mean pressure exerted on the screen across a subset of our full user population. To plot the figures, we computed each user's mean area (and mean pressure) and plotted the results on the CDF. Observe that more than 80% of the population had a mean area between 0.1 and 0.25 and that about 50% of the population had mean pressure values between 0.4 and 0.6. These user proportions



(a) Distribution of the mean pressure and mean area across touch strokes

(b) Distribution of the standard deviation of the mean pressure and mean area across touch strokes

Fig. 4. CDFs expressing the mean and variability of area and pressure seen across the population.

already suggest that a large number of users could be clustered around a narrow band of values (for both pressure and area). To get more concrete insight into the possible clustering of users' profiles, we studied the variability seen by users for each of these two variables.

Particularly, we computed the standard deviation of the mean area and mean pressure exhibited by each of the users represented in Figure 4(a) and then plotted these values on a CDF (Figure 4(b)). Taking the case of pressure, for instance, the figure shows that about 40% of the users had a standard deviation greater than 0.15. Assuming that users' mean pressure values follow a Gaussian distribution, a user with standard deviation of 0.15 could see his or her biometric pattern fall on a band having a width up to 0.6 units (i.e., 2 standard deviations on either side of the mean). Given such a wide span, an input selected from the earlier mentioned clustered regions (Figure 4(a)) could have a good chance of falling within such a user's feature range.

Similar observations made for the other features (e.g., velocity, length of strokes, start-point of stroke) further prompted us to hypothesize that generic information from the population could possibly enable us to implement a successful attack on a subset of the users.

4.3. Mechanical Design of the Robot

Figures 5 and 6 illustrate the mechanical design of the robot. Figure 5 shows how the robot components are connected, whereas Figure 6(a) and (b) respectively show the robotic "finger" used to touch the screen and a closer view of how some of the motors are connected. The piece of Play-Doh [Walsh 2005] at the bottom of the finger simulates the softness of a human finger, whereas the AA battery connected to it helps to increase the extent of electrical contact between the finger and the screen. Increased electrical contact between the finger and the screen enables the effective finger area and pressure on the screen registered by the Android system to be fairly high without having to push the finger so firmly against the screen (which in turn would complicate the swiping process and potentially deform the Play-Doh).

Motor C drives a network of gears that move the robotic finger vertically on and off the phone screen (see Movement C). As Motor C drives the finger vertically, the motor (i.e., Motor C) is itself also driven along a pair of rails (see Movement A) by Motor A. The resultant motion of the finger is hence a superposition of movements due to the Motors A, B, and C.

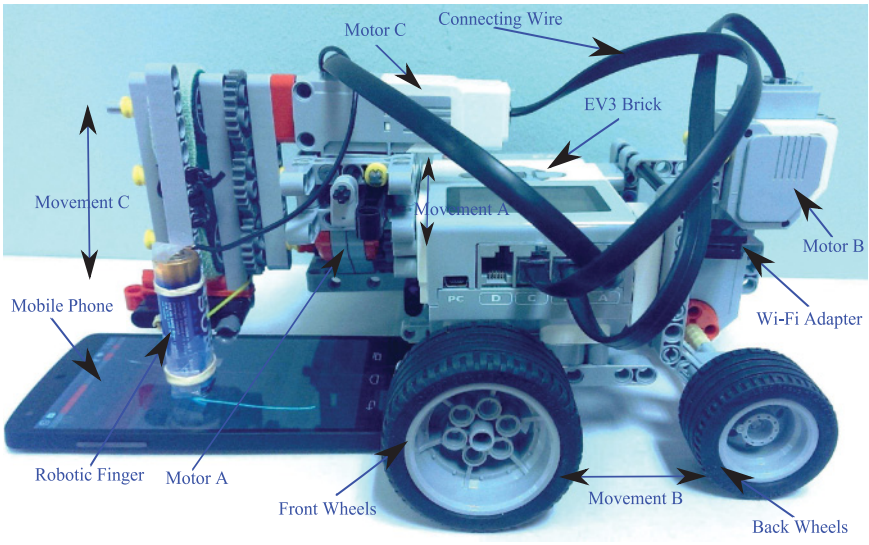
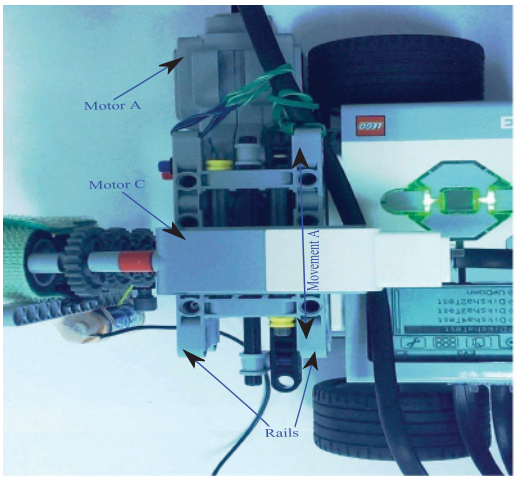
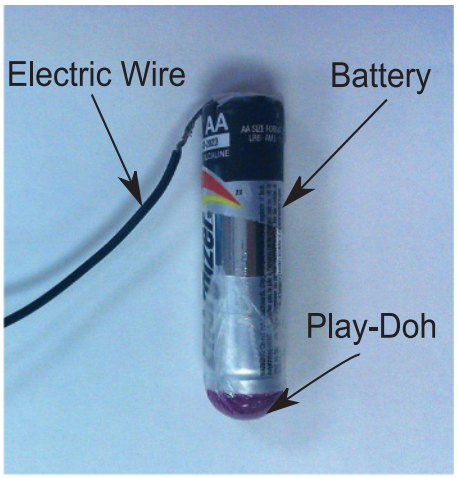


Fig. 5. Mechanical design of the robot. Motor C moves the robot vertically on and off the phone screen, whereas Motors A and B control the trajectory of the touch stroke as illustrated later in Figure 7. The parameters passed to each of these motors control attributes such as the speed, pressure, and length of a stroke.



(a) Motor C viewed from the top



(b) Design of robotic finger

Fig. 6. Zooming in on the robot parts.

4.4. Algorithmic Design of Robot

The trajectory of the stroke is determined by Motors A and B, which move the robot (i.e., the robotic finger in particular) in the plane of the phone screen. Figure 7 illustrates how a touch stroke is generated. The basic design is that Motors A and B move approximately at right angles to each other (see Movement A and Movement B in Figure 5) to create a stroke whose trajectory is the result of the two motor movements. Each motor's movement is controlled based on two parameters: the average speed of movement

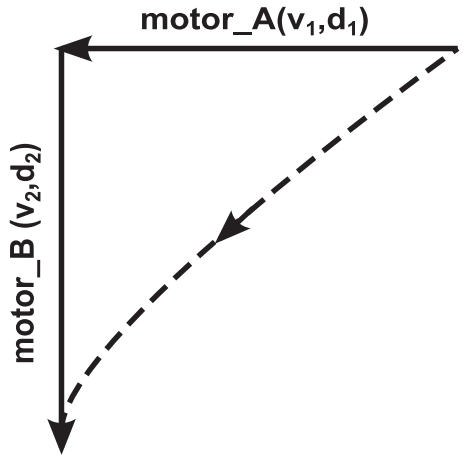
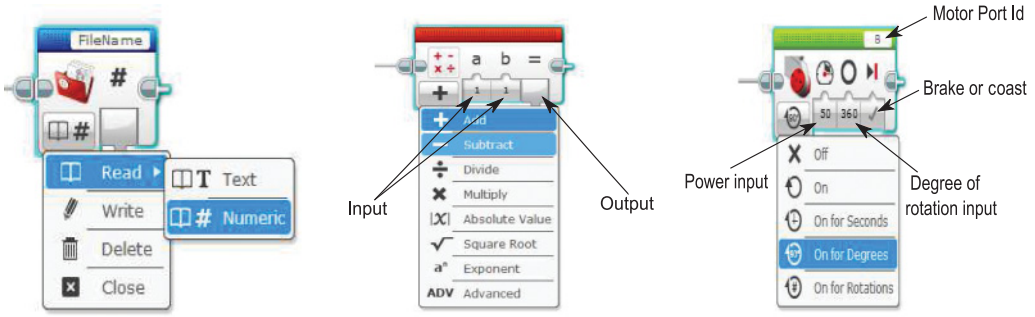


Fig. 7. How the motors generate a touch stroke. Two motors move concurrently at right angles to each other to generate a stroke (approximately) represented by the dashed line.

(e.g., v_1 for Motor A) and the distance moved (e.g., d_1 for Motor A). The values of these parameters determine the basic properties of the generated stroke, such as the stroke length and angular orientation and the finger speed along the stroke trajectory. In Figure 7, the dashed line represents a stroke that is generated when Motor A (whose trajectory is represented by the horizontal line) and Motor B (whose trajectory is represented by the vertical line) move concurrently in the directions shown in the figure. The stroke is not straight, as the mechanical dynamics of the robot parts cause the directions of motion of Motors A and B to deviate from the perpendicular paths shown in the figure. This stroke curvature, however, enables our strokes to be more similar to the *generic* human stroke, which is itself not exactly straight. Algorithm 1 summarizes the full swiping process. The input at the top of the inputs list is the number of strokes to be executed, whereas the next six inputs are the earlier mentioned speed and distance parameters for the three motors (i.e., each pair of inputs is for one motor). The Lego system specifies the speed inputs as units of motor power (that range from 1 to 100) and the distance inputs in terms of the number of rotations. Here, rfs is a noise term that we add to the distance parameter for one of the motors to ensure that the generated strokes are not exactly the same.

The $FingerDown()$ method moves the robotic finger vertically on and off the phone screen, whereas the $FingerForward()$ and $FingerRight()$ methods run in parallel to move the motors as already described in Figure 7. At the end of a stroke, the method $FingerUp()$ is called to remove the robotic finger off of the phone before the methods $FingerBackward()$ and $FingerLeft()$ run in parallel to return the robotic finger to the start point of a stroke after a short wait interval.

Figure 8 shows the GUI components used to implement Algorithm 1. An operation such as the forward component of finger motion due to Motor B is implemented based on the combination of the three blocks. For instance, input from the File Access block is passed to the Arithmetic Operation block, which manipulates this input (e.g., by adding a noise term) to produce a result that is passed to the Action block, which causes the movement of the motor. Reverse movement of the same motor calls for a similar set of blocks. The complete implementation of Algorithm 1 is hence a large network of blocks that we do not show here due to space limitations.



(a) File access block (b) Arithmetic operation block (c) Action block

Fig. 8. Some of the core GUI components that we used to program the robot. A File Access block specifies a file from which input parameters are read. The Arithmetic Operation block performs arithmetic manipulations on the input. The Action block executes the motor movement operations. On this block, the Motor Port Id specifies the motor (e.g., Motor B) that the block controls, whereas the power input and degree of rotation parameters respectively correspond to the speed and distance inputs that were discussed during our description of Figure 7. The full network of blocks required to implement Algorithm 1 is quite large and not shown here due to space limitations.

ALGORITHM 1: Swiping Mechanism

Input: numOfSwipes//`Number of strokes`;
Input: lenLeftRight//`Swipe length left-right`;
Input: speedLeftRight//`Speed left-right`;
Input: lenFwdBkwd//`Swipe length forward-backward`;
Input: speedFwdBkwd//`Speed forward-backward`;
Input: lenUpDown//`Finger up-down length`;
Input: speedUpDown//`Finger up-down speed`;
GenerateSwipes()
for $i \leftarrow 1$ **to** *numOfSwipes* **do**
  rfs = GRandom(μ_1,σ_1); `//Gaussian noise in swipe length`
  `//Creating swipe`
  FingerDown(speedUpDown, lenUpDown);
  FingerForward(speedFwdBkwd, lenFwdBkwd+rfs);
  FingerRight(speedLeftRight, lenLeftRight);
  `//Setting finger back to the original position`
  FingerUp(-speedUpDown, lenUpDown);
  FingerBackward(-speedFwdBkwd, lenFwdBkwd+rfs);
  FingerLeft(-speedLeftRight, lenLeftRight);
  Wait(swipeInterval);//`Inter-swipe interval`
end

4.5. Algorithm Inputs and Attack Types

4.5.1. Computing the Inputs. The two inputs to each motor (see Algorithm 1) are calculated based on simple arithmetic. Assume a scenario where the robot is to be used to forge the stroke represented by the dotted line in Figure 7 for a phone having a known pixel density. The distance input, d_1, for Motor A (which is essentially the number of rotations for Motor A; see Section 4.4) is computed as the length of the horizontal line (in pixels) in Figure 7 divided by the length (in pixels) that Motor A drives the robotic finger in one rotation. Having calculated the distance input, we proceed to divide it by the time taken[1] along the stroke to obtain the speed, V_1^*. Recall that the speed

[1]This time is gleaned from population statistics or user-specific data.

Table III. Input Parameters for Population Attack

Input	Portrait Swipe		Landscape Swipe	
	Horizontal	*Vertical*	*Horizontal*	*Vertical*
lenFwdBkwd	68	67	63	50
lenLeftRight	37	25	14	23
speedFwdBkwd	49	25	49	25
speedLeftRight	26	10	11	10

input to the Lego is in terms of a power metric ranging from 1 to 100. We hence do not directly input V_1^* to the algorithm. To convert V_1^* to units of power, we apply a simple calculation of ratios. In other words, if a speed input of 100 units gives a speed of α rotations per unit time, then our speed V_1^* can be converted to v_1 (which is the speed in units of power) using the following expression: $v_1 = \frac{100 * V_1^*}{\alpha}$. The same logic is used to compute the distance and speed inputs for Motor B. We next discuss the attack types and how we apply the preceding formulation to each one of them.

4.5.2. Population Statistics–Based Attack. In this attack, the adversary is assumed to have access to a large repository of touch biometrics data (e.g., publicly accessible datasets such as the one used in Frank et al. [2013]). Based on this data, the attacker identifies common traits across a population (e.g., typical stroke velocity and pressure of the common user) and then formulates the attack such that its inputs translate into robotic strokes that match the observed common traits. For this attack, our algorithmic input was based on mean values observed across a small section of the population that was specifically dedicated to tuning the attack.

Using the mean start-point and mean end-point of touch strokes across this subpopulation, we computed the horizontal and vertical distances represented in Figure 7, which we then used in the formulation described in Section 4.5.1. Further, we computed the mean time taken along a stroke for this population and used it to compute the speed input as described in the formulation. For each of landscape-vertical, landscape-horizontal, portrait-vertical, and portrait-horizontal strokes, we computed separate sets of inputs. Table III shows the algorithm inputs for each type of stroke. Note that these inputs are computed based on the Samsung Google Nexus S phone that has a pixel density of 233 pixels per inch. The tabulated values of *lenFwdBkwd* and *lenLeftRight* are in degrees (as opposed to number of rotations) and were computed based on wheels of diameter 1.6 inches.

For the pressure and area inputs, we molded the Play-Doh lamp at the tip of the robotic finger to produce a pressure between 0.4 and 0.6 units and an area of 0.15 units, which were the mean values observed across the earlier mentioned training subpopulation that was dedicated to tuning the attack. This process of molding the Play-Doh was undertaken during a set of trial and error experiments run before the attack. For each of the four stroke categories, we placed the robotic finger at a point corresponding to the mean start-point observed for the training subpopulation before starting the robotic swiping process.

4.5.3. User-Tailored Attack. In this attack, the adversary designs the robot to specifically mimic a given user's swiping pattern. To be able to do this, the adversary is assumed to have somehow accessed the intended victim's swiping samples before gleaning information from them to train the robot. Such an attack can be launched by an insider attacker who is interested in retrieving high-value information on the phone of a person he or she knows so well and has access to. Consider a scenario in which Alice wants to access private information from Bob's phone. A simple way in which Alice could access Bob's swiping data is by asking Bob to browse some pages (pages

containing some interesting images, tweets, etc.) on Alice's phone while an application in the background captures Bob's swiping behavior.

In the long run, Alice would then either (1) wait for a moment when Bob leaves his phone unattended for some time or (2) outrightly steal Bob's phone so as to use the robot to overcome the touch-based authentication system on the phone. In the event that Alice is unable to access Bob's phone, she could advertise his swiping pattern in underground networks where another adversary could target Bob to get the phone.

To generate algorithmic input for this attack, we use the same formulation that was described in Section 4.5.1 and applied to the population attack. The only variation from the population attack is that while algorithm inputs were calculated based on population statistics under the population attack, this time around they are computed based on the victim's own data. For this attack, we hence do not tabulate the set of inputs, as each user subjected to the attack had his or her own set of inputs.

5. VERIFICATION ALGORITHMS AND PERFORMANCE EVALUATION METHODOLOGY

5.1. Verification Algorithms Used

We used the following classification algorithms to evaluate the impact of the attacks: naive Bayes [Duda et al. 2002], logistic regression [Witten and Frank 2005], SVM [Cortes and Vapnik 1995], kNN [Cover and Hart 2006], random forests [Breiman 2001], Bayesian networks [Duda et al. 2002], and neural networks [Duda et al. 2002] (i.e., multilayer perceptron). These algorithms are known to address both linear and nonlinear recognition problems (e.g., SVM is suited for both linear and nonlinear recognition problems, whereas kNN, neural networks, and random forests are for nonlinear problems), and they have different operational philosophies (e.g., naive Bayes is generative, logistic regression is discriminative, and random forests is induction and ensemble based). This wide range of algorithms will help provide a rigorous understanding of the impact of the attack on different classification methodologies. Due to space limitations, we do not discuss the operational mechanisms of these algorithms, as they are widely used and well understood in the machine learning community.

5.2. Training and Testing Method

For each user, data collected during the first session was used for training, whereas data collected during the second session was used for testing (recall data collection sessions described in Section 3.1). For reasons already given in Section 3.2, each user's strokes were categorized as either portrait-vertical, portrait-horizontal, landscape-vertical, or landscape-horizontal. For each of these stroke categories, training was done to build a user's model, with testing later done to assess the user's performance for each stroke category. Training (for each category of stroke) was done using at least 80 strokes for each user (i.e., users who had fewer than 80 strokes were excluded from the analysis). We fixed this minimum number of strokes because we found that classification performance suffered significantly when user models were built based on much smaller numbers of strokes.

For each user's four training models (i.e., templates), we carried out three types of tests: one in which the user's own strokes were matched against the template (we also refer to this as the genuine attack), one in which strokes drawn from random impostors were matched against the template (we also refer to this test as the zero-effort attack), and one in which strokes generated by the robot were matched against the template (we refer to this test as the robotic attack[2]). For the genuine attack, we used 80 strokes

[2]The robotic attack is further subdivided into the user-specific and population statistics–based attack, each of which has a separate test.

executed by the user in question, whereas for the zero-effort attack, we used 10 strokes from each of 30 randomly selected impostors. To evaluate the robotic attack, we used 300 strokes generated by the robot.

For both template building and the three kinds of tests, each feature vector was formulated from a window of 10 consecutive touch strokes (as opposed to a single touch stroke). The single vector was computed such that its elements were the component-wise means of the elements of the 10 (28-dimensional) vectors comprising a window (refer to Section 3.1 for details of the vector elements). Because a user will every now and then execute a stroke that is distinct from the rest of her strokes, the combination of multiple strokes—also used in Frank et al. [2013] and Li et al. [2013]—helps to formulate a more coherent representation of a user's touch behavior than the use of a single stroke. For each additional stroke in excess of the 10 strokes comprising a window, the window slides forward to contain the 10 latest touch strokes.

5.3. Performance Evaluation Approach

We evaluate the impact of the robotic attack based on the increase in FAR caused by the attack. The FAR is the ratio of the number of impostor attempts that get accepted by the system to the total number of impostor attempts input to the system. To compute the FAR, one must first set a classification threshold based on which a given sample is accepted or rejected by the system. We use the EER threshold, as it is widely used in behavioral biometrics in general and in touch-based authentication in particular (e.g., see Frank et al. [2013], Killourhy and Maxion [2009], and Govindarajan et al. [2013]). The EER is the error rate of an authentication system when the FAR equals the FRR; the EER threshold is the verification score at which the EER is obtained.

We first compute the EER (and its associated threshold) when the system is being used normally (i.e., before the robotic attacks are launched) and then compute the FAR based on the same threshold after the robotic attack is launched. Recall that by definition, $EER = FAR = FRR$ at the EER threshold, which implies that the previously mentioned EER computed before the robotic attack is equivalent to the FAR of the system before the attack is launched. This in turn implies that the change in FAR due to the attack can simply be computed by subtracting the EER before the attack from the FAR obtained after the attack. Since each user has four distinct biometric profiles (review Section 3.2 for the rationale behind this design choice), we compute four different thresholds for each user and evaluate the attack separately for each category of strokes.

6. ATTACK RESULTS

6.1. Baseline Results

Here we discuss the performance of our system before the robotic attack was launched (i.e., when legitimate users' data was input to it). We refer to the system's performance under these conditions as the *baseline* performance of the system. If our baseline performance can be found to be comparable to the performance seen with existing continuous authentication systems, it is reasonable to hypothesize that our attack results should to a good extent be reflective of the performance of other systems in the literature. Table IV summarizes the baseline performance of our system (see two columns under the heading *Baseline*). The table shows that our lowest FAR at baseline (recall that this was equivalent to the EER; see Section 5.3) was 17%. This EER/FAR is much higher than the error rates reported in some works (e.g., ≈4% reported in Frank et al. [2013]) but is comparable to the error rates reported in several other papers (e.g., FARs of just under 15% reported in Feng et al. [2012] in the experiments where users did not wear a glove, whereas EERs of about 20% are reported in Govindarajan

Table IV. Impact of the Population Attack on the Classification Performance of the Horizontal and Vertical Portrait Strokes

Classifier	Portrait Horizontal Strokes					
	Baseline		*Population Attack*			
	μ_{FAR} Before Attack	σ_{FAR} Before Attack	μ_{FAR} After Attack	σ_{FAR} After Attack	Increase in μ_{FAR} (%)	Increase in σ_{FAR} (%)
log-reg	0.17	0.17	0.42	0.48	143	130
knn	0.18	0.16	0.41	0.47	130	198
ran-for	0.18	0.18	0.48	0.39	168	121
svm	0.19	0.19	0.56	0.46	189	145
bay-net	0.21	0.20	0.58	0.38	181	94
mul-per	0.22	0.24	0.64	0.41	182	73
nai-bay	0.23	0.24	0.60	0.38	159	57
	Portrait Vertical Strokes					
log-reg	0.22	0.17	0.43	0.46	95	167
knn	0.31	0.19	0.52	0.47	71	148
ran-for	0.28	0.23	0.49	0.46	170	96
svm	0.23	0.17	0.51	0.47	126	172
bay-net	0.32	0.24	0.64	0.47	102	93
mul-per	0.27	0.24	0.62	0.46	129	88
nai-bay	0.32	0.27	0.67	0.47	108	73

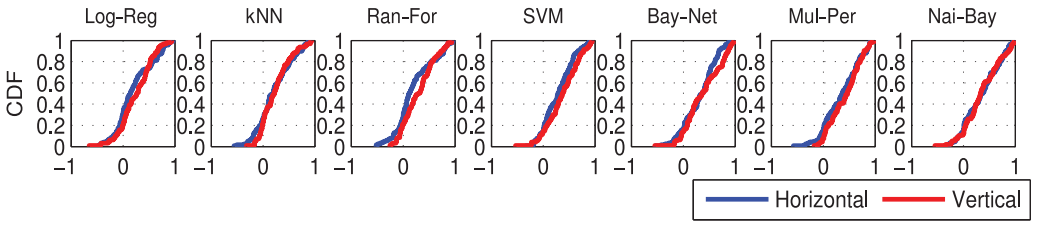


Fig. 9. Impact of the population attack on the classification performance of each user for the vertical and horizontal portrait strokes.

et al. [2013]). The differences in performance across studies are likely due to a wide range of reasons (e.g., differences in study populations, differences in the specific tasks undertaken by the users during data collection) that are difficult to standardize. What is crucial for our investigations, however, is that our system is comparable to several systems in the literature. In the following section, we discuss how the attacks change the baseline performance of the system.

6.2. Population Attack

6.2.1. Impact of the Attack with All Users Allowed to Enroll. Table IV and Figure 9 show the impact of the attack when all users in our study were allowed to enroll. We focus here on the horizontal and vertical portrait strokes because the other kinds of strokes gave no new insights. Table IV shows the mean EER and standard deviation of the EERs across the population before and after the population-based robotic attack and the percentage change in these two variables as a result of the robotic attack. The table helps to capture the global impact of the attack. Figure 9, on the other hand, captures the effect of the attack on each user. Observations on Table IV and Figure 9 are summarized next:

(1) *Increased mean FAR*: The attack caused a drastic increment in the mean EER across all seven verifiers. For example, the logistic regression verifier had an FAR of 0.17 before the robotic attack, which was more than doubled to 0.42 as a result of the attack. A similar trend is seen for the other verifiers and for both kinds of strokes. This increment in mean FAR shows that an impostor using a robot would see much higher success rates than the naive impostor who makes no effort to forge swiping patterns.
(2) *Increased standard deviation of FAR*: The table also shows that the standard deviation of the FARs across the population increased remarkably under the attack. For example, for the logistic regression verifier with the portrait-horizontal strokes, the standard deviation of the FAR increased by 130% as a result of the attack. For both types of strokes, the naive Bayes verifier had the lowest increment in the standard deviation of the FAR (57% and 73% for the horizontal and portrait vertical strokes, respectively); however, the impact of the attack is still clear. In general, a high standard deviation of the error rates implies that the system becomes unpredictable as a result of the attack.
(3) *Some users are immune to the population-based attack*: Figure 9 shows the impact of the attack on each user's classification performance. To plot the figure, we subtract each user's FAR before the attack from that after the attack and then plot a CDF of the differences. As seen from the figure, there are certain users who had differences less than zero, implying that their FARs were lower after the attack. This phenomena, also reported for the population-based attack on the keystroke authentication system in Serwadda and Phoha [2013a], shows that there is a certain group of users whose behavior is distinct from that of the population. For such users, patterns gleaned from the general population cannot be used as a basis for a successful attack. For each of the verifiers, the proportion of users depicting this trait was between 20% and 40% of the population.
(4) *Change in verification algorithm has no marked impact on attack*: All seven verification algorithms were markedly negatively impacted by the attacks. The lowest increment in mean FAR due to the attack was about 71% (see the kNN verifier for the portrait vertical strokes), which is a very high percentage of the baseline FAR. Overall, this means that changes in the philosophy behind the classification engine cannot help to thwart the attack.

6.2.2. Impact of a Failure-to-Enroll Policy. To more rigorously explore the effect of the attack, we employed a failure-to-enroll policy and excluded all users whose baseline FAR was greater than a certain threshold α. The logic behind this exclusion of certain users is that for users who have a very high FAR at baseline (i.e., under the zero-effort attack), there is likely no need to use a robot to break their touch patterns given that a nonzero-effort attack is already highly successful. By eliminating these users, we are hence able to analyze the effect of the attack on the most consistent users of the system. Figure 10 and Table V summarize the impact of our attack at different failure-to-enroll thresholds.

The threshold $\alpha = 1$ (see Figure 10) means that no user who met the earlier described 80 strokes requirement (see Section 5.2) was denied enrollment, whereas $\alpha = 0.4$ means that all users whose mean FAR[3] at baseline was above 0.4 are excluded from the evaluation. The former case (i.e., $\alpha = 1$) is included for comparison purposes. For each value of α, Figure 10 also shows the number of users who are able to enroll (e.g., at $\alpha =$ 0.4, 92 users were able to enroll).

[3]This mean FAR is computed based only on the four verification algorithms having the lowest mean FAR at baseline.

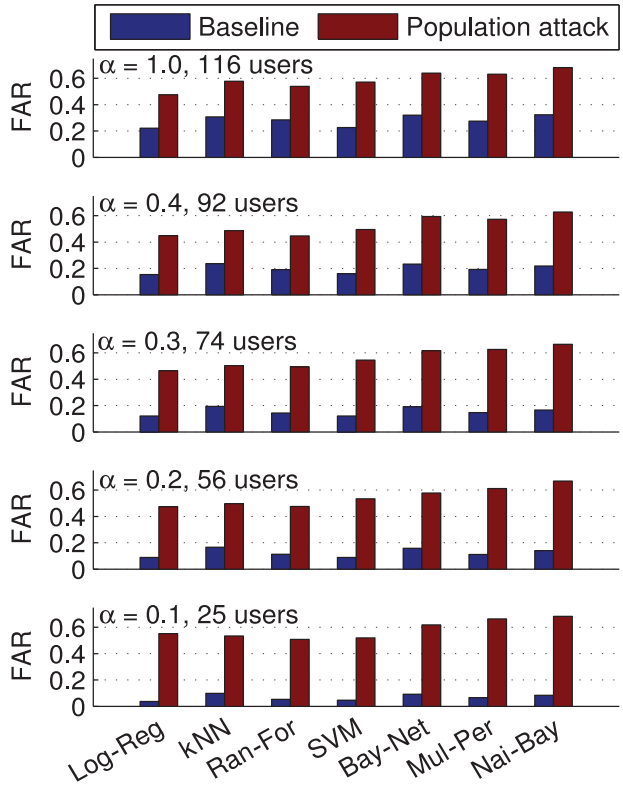


Fig. 10. Effect of the population attack at different failure-to-enroll thresholds.

Table V. Effect of a Failure-to-Enroll Policy on the Performance of the Population Attack with the Portrait Vertical Touch Strokes

Classifier	$\alpha = 0.4$					
	Baseline		*Population Attack*			
	μ_{FAR} Before Attack	σ_{FAR} Before Attack	μ_{FAR} After Attack	σ_{FAR} After Attack	Increase in μ_{FAR} (%)	Increase in σ_{FAR} (%)
log-reg	0.15	0.12	0.55	0.48	260	349
knn	0.24	0.11	0.52	0.44	119	237
ran-for	0.19	0.11	0.50	0.47	164	250
svm	0.16	0.14	0.53	0.48	236	326
bay-net	0.23	0.12	0.57	0.43	144	189
mul-per	0.19	0.20	0.62	0.40	227	154
nai-bay	0.22	0.17	0.66	0.42	205	181
	$\alpha = 0.1$					
log-reg	0.04	0.09	0.54	0.41	1,387	1330
knn	0.10	0.10	0.48	0.35	397	466
ran-for	0.05	0.10	0.47	0.43	808	1,011
svm	0.05	0.10	0.47	0.43	936	1,152
bay-net	0.09	0.11	0.52	0.33	464	425
mul-per	0.07	0.17	0.66	0.37	900	681
nai-bay	0.08	0.13	0.69	0.38	730	468

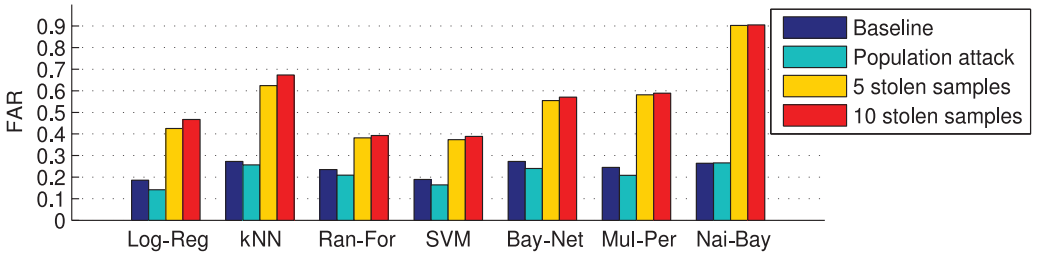


Fig. 11. Effect of user-specific robotic attack for different amounts of stolen swiping data.

Figure 10 shows that even with a failure-to-enroll policy in place, the attack is still effective. The figure is such that the first column of bars corresponds to the logistic regression verifier (Log-Reg), the second column of bars corresponds to the kNN verifier, and so on. Taking the case of $\alpha = 0.3$, for example, the figure shows that all seven verifiers have their mean FAR equal to or greater than 0.4 after the robotic attack, whereas none of them had a mean FAR exceeding 0.2 before the robotic attack was launched. This is equivalent to more than a 100% increment in mean FAR for each of the verifiers as a result of the attack.

Table V gives the exact percentage increments in mean FAR for the thresholds $\alpha = 0.4$ and $\alpha = 0.1$. The table also shows the change in standard deviation of the FARs due to the attack. We only show results for the portrait-vertical strokes and the two thresholds because the other kinds of strokes and thresholds gave no notable new insights. Observe that both the mean FARs and standard deviations of FARs increased drastically for all verifiers, confirming that the attack is still effective even when the poor-performing users are barred from enrolling onto the system.

6.3. User-Tailored Attack

We finally evaluate the effect of the user-specific form of the robotic attack on the subset of users who were most resistant to the population-based attack. We selected 30 users who were most resistant to the population attack and subjected their profiles to the user-specific attack. By *most resistant*, we mean the users whose FARs decreased, or those who had the lowest increments in mean FAR as a result of the population attack. Our assumption is that if a given user's profile easily succumbs to the population attack, an attacker would have no motivation to carry out the more intricate user-specific attack for that user. It is for this reason that we focus on the subset of users who had the lowest FAR increment under the population attack.

We vary the amount of data stolen by the attacker between 1, 5, and 10 touch strokes. Choice of these three small (stolen) sample sizes is due to the assumption that the attacker is unlikely to get the victim to perform a great deal of swiping.

Figure 11 summarizes the impact of the user-specific attack in comparison to the zero-effort attack and the population attack. For each set of bars on the figure, the first bar on the left corresponds to the zero-effort attack, the second bar corresponds to the population attack, and the last two bars represent the user-specific attack with the attacker using 5 and 10 stolen samples, respectively. The stolen samples are randomly selected. Observe that for some cases, the mean FAR under the population attack is less than the baseline FAR under the zero-effort attack. This is because the EERs plotted on this graph are computed based on the earlier mentioned subpopulation of *resistant* users for whom the population attack sometimes caused lower FARs than the zero-effort attack.

In comparison to both the zero-effort and population attacks, Figure 11 shows that the user-specific attack causes the system to attain much higher mean FARs for all seven verification algorithms. Interestingly, stealing 10 samples does not appear to produce significant performance gains from the attacker's perspective relative to when just 5 samples are used. This implies that 5 samples already provide a significant amount of information about the user.

7. DISCUSSION AND CONCLUSIONS

In this section, we explore some of the weaknesses of our attack design and performance evaluation methodology before presenting our conclusions.

7.1. Discussion

Attack performance evaluation. Although the system to which we subjected our attacks (i.e., the baseline authentication system) is comparable to and in some cases performs better than several systems (in terms of classification error rates) that have recently been proposed in the literature (see Sections 2.1 and 6.1), there are studies that have showcased systems with error rates lower than our baseline error rates. These differences in error rates across independent studies emanate from a range of factors (e.g., user composition, feature sets, experimental conditions, i.e., a specific use case for which users provide data, constraints imposed on users as they provide data, phone screen resolutions, data sampling rates) that are difficult to precisely standardize across different studies. One weakness of our performance evaluation methodology is that we were unable to compare the performance of our attacks across a wide range of previously proposed systems.

For some of the previously proposed systems, it is possible that our attacks would not perform as well as they performed in our experiments. This dip in attack performance, however, is more likely for the population statistics–driven attack (see Section 6.2), which heavily relies on the overlap across user profiles. Our belief is that the user-tailored attack—by virtue of closely mimicking the victim's swiping behavior—would still cause a significant performance degradation irrespective of the system being evaluated.

Patterns in swiping behavior. In Section 4.2, we made observations on users' swiping patterns (e.g., regions of swiping activity, common values of pressure). In practice, these swiping patterns should be influenced by the design of the user interface of the application under consideration. For example, if one used an app that has buttons on the entire bottom half of the screen with text occupying only the top half of the screen, swiping activity would potentially be concentrated more in the region having text than the region having the buttons (i.e., users would likely browse in the region not containing the buttons to avoid inadvertently clicking the buttons). This observation implies that if one were to capture swiping patterns using an app whose interface or nature of operation was markedly different from the one used in our work, it is possible that patterns different from those seen in our work would be observed.

That said, as described in Section 3.1, our experiment very closely simulated the standard browsing process that is seen with many apps in which users have to read and process content on the screen (e.g., Web browsing, reading e-mails). In particular, as users searched for solutions to the multiple choice questions, they had to carefully browse the page while reading and processing text that contained the clues. Our belief is that our observations should be representative of a large subset of smartphone apps

that involve reading text,[4] and thus could be used for the design of attacks on a broad range of apps in this family.

Irrespective of whether our observed patterns apply to a given app or not, a more interesting option for the attacker is to exploit data obtained from the app that is to be attacked so as to glean patterns that precisely represent this app. For example, if the attacker intends to break into a banking app, he or she would have to glean patterns seen with people using this very app. In practice, this should not be so difficult, as the attacker can use data provided by accomplices (e.g., see Ballard et al. [2006] and Serwadda and Phoha [2013a]) or data from unsuspecting users (who may be fooled into browsing on the attacker's phone).

Mechanical design issues. Another aspect that manifests as a weakness of our attack design is the fact that the mechanical design of the Lego is in a way "hard coded" to the shape and size of the phone. In particular, the identities of the Lego parts, their orientation, sizes, and mechanism of interaction were all very tightly coupled to the device that was attacked in our study. In practice, this implies that for the adversary to launch the attack against a device such as a touchscreen laptop, a new mechanical design that meshes well with the mechanical dynamics of the laptop (e.g., bigger screen, laptop keyboard occupying space in front of the keyboard) would need to be implemented. A general-purpose form of the attack would require a more sophisticated robot (e.g., a humanoid robot) that only requires programmatic manipulation as one switches from one victim device (e.g., smart phone) to another (e.g., touchscreen TV).

7.2. Conclusions

In this article, we have presented a population-based and a user-specific robotic attack on touch-based authentication. For a set of seven verification algorithms having baseline FARs of between 0.17 and 0.32, we have shown the population attack to cause an FAR increment of greater than 70% for the least affected algorithm. For the user-specific attack, we have shown that the effect of the attack is even much more drastic, with the baseline FAR of the least affected algorithm being multiplied over twofold as a result of the attack.

Because the attacks require only basic programming skills and are launched using cheap off-the-shelf hardware, they represent a realistic threat that should be expected to be faced by a real deployment of a touch-based authentication system. The article not only calls for the incorporation of robotic attacks in the standard impostor testing routine of touch-based authentication systems but also calls for research into mechanisms that could defeat these attacks. One appealing approach to fortify touch-based authentication is to leverage information from the multitude of sensors built in smart phones these days to implement some kind of fusion framework that identifies a user based on multiple modalities. For example, using the motion and inertial sensors in the phone, such a system could capture the signature of the subtle movements of the phone while a user swipes and then use them in conjunction with the swiping signature to identify the user. Although the use of mechanical devices (e.g., robots) to forge this fortified signature might not be completely impossible, it could make the attacker's work significantly more difficult.

Another possible defense against these kinds of attacks is the use of pattern recognition techniques (i.e., liveness tests) that can delineate between the swiping pattern of a human and that of a robot. The principal idea behind this kind of defense is that robotic movements rotate around a small set of canonical movements. For example,

[4]Recall that the primary use case of our attack is for the adversary to read/browse sensitive private information on the phone without being detected by the touch-based authentication system.

a given motor could rotate left or right through one of a finite set of possible angles, and the full combination of all motors could only be able to execute a certain set of movements at a given point in time. Humans, on the other hand, are able to execute much more complex movements (many possibilities of angles, step sizes of rotation, etc.). By analyzing a presented pattern for certain predictable robotic behavior, one could determine the probability that the presented pattern is robotic. This option of defense, however, seems much more complex than the first, as different robots might have differences in operational dynamics. Thus, a defense that works against a certain family of robots mightnot work the same way against a different family of robots. Part of our future work will involve examining the various approaches to defending against these attacks.

REFERENCES

De Luca Alexander, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. 2012. Touch me once and I know it's you! Implicit authentication based on touch screen patterns. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems (CHI'12)*. ACM, New York, NY, 987–996. DOI:http://dx.doi.org/10.1145/2208516.2208544

Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith. 2010. Smudge attacks on smartphone touch screens. In *Proceedings of the 4th USENIX Conference on Offensive Technologies (WOOT'10)*. 1–7. http://dl.acm.org/citation.cfm?id=1925004.1925009.

Lucas Ballard, Seny Kamara, Fabian Monrose, and Michael K. Reiter. 2008. Towards practical biometric key generation with randomized biometric templates. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*. ACM, New York, NY, 235–244. DOI:http://dx.doi.org/10.1145/1455770.1455801

L. Ballard, D. Lopresti, and F. Monrose. 2007. Forgery quality and its implications for behavioral biometric security. *Transactions on Systems, Man, and Cybernetics, Part B* 37, 5, 1107–1118. DOI:http://dx.doi.org/10.1109/TSMCB.2007.903539

Lucas Ballard, Fabian Monrose, and Daniel Lopresti. 2006. Biometric authentication revisited: Understanding the impact of wolves in sheep's clothing. In *Proceedings of the 15th Conference on USENIX Security Symposium*, Vol. 15 *(USENIX-SS'06)*. Article No. 3. http://dl.acm.org/citation.cfm?id=1267336.1267339.

R. M. Bolle, S. Pankanti, and N. K. Ratha. 2000. Evaluation techniques for biometrics-based authentication systems (FRR). In *Proceedings of the 15th International Conference on Pattern Recognition*, Vol. 2. 831–837 DOI:http://dx.doi.org/10.1109/ICPR.2000.906204

Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1, 5–32. DOI:http://dx.doi.org/10.1023/A:1010933404324

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3 , 273–297. DOI:http://dx.doi.org/10.1023/A:1022627411411

T. Cover and P. Hart. 2006. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13, 1, 21–27. DOI:http://dx.doi.org/10.1109/TIT.1967.1053964

R. Duda, P. Hart, and D. Stork. 2002. *Pattern Classification* (2nd ed.). John Wiley & Sons.

Tao Feng, Liu Ziyi, Carbunar Bogdan, Boumber Daining, and Shi Weidong. 2012. Continuous mobile authentication using touchscreen gestures. In *Proceedings of the 12th IEEE Conference on Technologies for Homeland Security (HST'12)*.

Mario Frank, Ralf Biedert, Ma Eugene, Martinovic Ivan, and Song Dawn. 2013. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE Transactions on Information Forensics and Security* 8, 1, 136–148.

Mitchell H. Gail and Sylvan B. Green. 1976. Critical values for the one-sided two-sample Kolmogorov-Smirnov statistic. *Journal of the American Statistical Association* 71, 355, 757–760.

S. Govindarajan, P. Gasti, and K. S. Balagani. 2013. Secure privacy-preserving protocols for outsourcing continuous authentication of smartphone users with touch data. In *Proceedings of the 2013 IEEE 6th International Conference on Biometrics: Theory, Applications, and Systems (BTAS'13)*. 1–8. DOI:http://dx.doi.org/10.1109/BTAS.2013.6712742

A. Rahman Khandaker, Kiran S. Balagani, and Vir V. Phoha. 2013. Snoop-forge-replay attacks on continuous verification with keystrokes. *IEEE Transactions on Information Forensics and Security* 8, 3, 528–541.

Kevin S. Killourhy and Roy A. Maxion. 2009. Comparing anomaly-detection algorithms for keystroke dynamics. In *Proceedings of the 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'09)*. 125–134.

Lingjun Li, Xinxin Zhao, and Guoliang Xue. 2013. Unobservable reauthentication for smart phones. In *Proceedings of the 20th Network and Distributed System Security Symposium (NDSS'13)*.

Elizabeth Mauch. 2001. Using technological innovation to improve the problem-solving skills of middle school students: Educators' experiences with the LEGO mindstorms robotic invention system. *Clearing House* 74, 4, 211–214.

Tey Chee Meng, Payas Gupta, and Debin Gao. 2013. I can be you: Questioning the use of keystroke dynamics as a biometric. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS'13)*.

Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. 2012. ACCessory: Password inference using accelerometers on smartphones. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications (HotMobile'12)*. ACM, New York, NY, Article No. 9. DOI:http://dx.doi.org/10.1145/2162081.2162095

Abdul Serwadda and Vir V. Phoha. 2013a. Examining a large keystroke biometrics dataset for statistical-attack openings. *ACM Transactions on Information and System Security* 16, 2, Article No. 8. DOI:http://dx.doi.org/10.1145/2516960

Abdul Serwadda and Vir V. Phoha. 2013b. When kids' toys breach mobile phone security. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS'13)*. ACM, New York, NY, 599–610. DOI:http://dx.doi.org/10.1145/2508859.2516659

Abdul Serwadda, Vir V. Phoha, and Zibo Wang. 2013. Which verifiers work? A benchmark evaluation of touch-based authentication algorithms. In *Proceedings of the 2013 IEEE 6th International Conference on Biometrics: Theory, Applications, and Systems (BTAS'13)*. 1–8. DOI:http://dx.doi.org/10.1109/BTAS.2013.6712758

Umut Uludag and Anil K. Jain. 2004. Attacks on biometric systems: A case study in fingerprints. In *Proceedings of SPIE5306: Security, Steganography, and Watermarking of Multimedia Contents VI*. 622–633.

Tim Walsh. 2005. *Timeless Toys: Classic Toys and the Playmakers Who Created Them*. McMeel Publishing.

Z. Wang, A. Serwadda, K. S. Balagani, and V. V. Phoha. 2012. Transforming animals in a cyber-behavioral biometric menagerie with frog-boiling attacks. In *Proceedings of the 2012 IEEE 5th International Conference on Biometrics: Theory, Applications, and Systems (BTAS'12)*. 289–296. DOI:http://dx.doi.org/10.1109/BTAS.2012.6374591

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques* (2nd ed.). Morgan Kaufmann, San Francisco, CA.

Xi Zhao, Tao Feng, and Weidong Shi. 2013. Continuous mobile authentication using a novel graphic touch gesture feature. In *Proceedings of the 2013 IEEE 6th International Conference on Biometrics: Theory, Applications, and Systems (BTAS'13)*. 1–6. DOI:http://dx.doi.org/10.1109/BTAS.2013.6712747

Received March 2015; revised December 2015; accepted February 2016