On Humanoid Robots Imitating Human Touch Gestures on the Smart Phone

Conference Paper · September 2015

DOI: 10.1109/BTAS.2015.7358781

3 authors, including:

Sujit Poudel
Syracuse University
3 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Vir V. Phoha
Louisiana Tech University
216 PUBLICATIONS **3,236** CITATIONS
SEE PROFILE

On Humanoid Robots Imitating Human Touch Gestures on the Smart Phone

Sujit Poudel Abdul Serwadda Vir V. Phoha
Syracuse University, Syracuse, NY 13210
{spoudel,aserwadd,vvphoha}@syr.edu

Abstract

We showcase an attack in which an autonomous humanoid robot is trained to execute touch gestures that match those of a target user. Different from past work which addressed a similar problem using a Lego robot, we harness the significant processing power and unique motoric capabilities of the autonomous humanoid robot to implement an attack that: (1) executes touch gestures with high precision, (2) is easily adapted to execute gestures on different touch screen devices, and (3) requires minimal human involvement. Relative to the traditional zero-effort impostor attacks, we show, based on a dataset of 26 users, that our attack significantly degrades the performance of touch-based authentication systems. In addition to the paper highlighting the threat that sophisticated adversaries pose to touch-based authentication systems, our robotic attack design provides a blueprint for much needed impostor testing mechanisms that simulate algorithmic (or sophisticated) adversaries against touch-based authentication systems.

1. Introduction

The past few years have seen a surge in interest in sensor-driven user authentication on smart phones. One of the most studied sensors has been the touch sensor, with researchers showcasing a wide range of mechanisms by which it can be leveraged to authenticate users based on patterns seen in their touch gestures (e.g., see [9][14][8][13]). Research on this problem has however been lopsided. While there is a huge volume of literature on how different authentication approaches work on touch biometrics data, there is surprisingly very limited research on the kinds of sophisticated (e.g., algorithmic) attacks that these systems could face if widely deployed. Without deep insights into the implementation dynamics and potential impacts of attacks that these systems could face in the wild, it is very difficult for researchers to design systems that can stand firm in the wake of adversity.

To our knowledge the only work to have demonstrated a sophisticated attack on touch-based authentication to date was our work in [15] where a Lego robot was used to mimic human touch gestures. That work however left several critical unanswered questions, two of which are briefly discussed next:

(1) *Attack Design Limitations*: With the simplicity of the attack being one of the core aims of the attack in [15], the paper used a very simple robotic device, and made several design choices aimed to simplify the execution of the attack. Partly due to the use of a robot with limited capabilities and partly due to the deliberate simplification of the design, the attack in [15] made significant compromises on the precision of execution of touch strokes. Further, the attack very heavily relied on the assumption that the Android OS records touch gestures using a very imprecise sampling resolution, was significantly limited in the shapes of gestures that could be executed and had components of it that required significant human involvement. As we discuss in Section 2.1, these issues leave behind a plethora of questions, many of which are addressed by the attack implementation in this work.

(2) *Attack Performance Evaluation*: Another question facing the attack in [15] regards the methodology used to evaluate the attack. Specifically, to demonstrate the impact of the robotic attack on the *continuous* touch-based authentication system, the paper solely used the traditional error metrics (e.g., False Acceptance Rate (FAR), Equal Error Rate (EER), etc.), demonstrating that the FAR (and hence the EER) seen during the robotic attack was higher than that seen when non-robotic impostors attacked the system. The challenge with using measures such as the EER *on their own* when characterizing the performance of a *continuous* authentication biometric system is that they do not convey any information about the temporal distribution of errors [7][16].

As an illustrating example of why this would be a problem, consider a robot that attains a given FAR over a session of touch gesture execution on the smart phone. Without any information on when the robot encounters failed authentication attempts during this session, it is very difficult to fully asses the level of threat that the robot poses to the system. If for instance the robot encounters consecu-

tive failed authentication attempts (i.e., bursts of rejections) during a given time window, it is very likely that the system will raise an alarm and lock out the robot. On the other hand if the robot has failed authentication attempts widely spread out, it is possible that the robot could have continued access to the system, since genuine users themselves also have occasional failed authentication attempts.

With interest in this area growing fast and federal agencies such as DARPA, AFRL and DHS working towards the development of sensor-driven smart phone authentication solutions (see relevant sections of recent proposal solicitations [1][3][2]), the deficiencies noted above in (1) and (2) highlight the need for a robotic impostor testing design for touch based authentication systems that not only implements touch strokes with high precision, but is also automated enough to be feasible for the implementation of large numbers of impostor tests on different kinds of devices with minimal human involvement. The work in this paper represents the first steps towards this end. In particular the paper makes the following contributions to the field:

1. We use an autonomous humanoid robot to design and implement an attack that mimics the human execution of touch gestures on a smart phone. Harnessing the significant processing power and unique motoric capabilities (relative to the Lego used in [15]) of the robot, we demonstrate how the robot can learn a given swiping behavior from empirical data and execute strokes that very closely match those of a human.

2. To rigorously evaluate the impact of the attack on the authentication system, we go beyond the traditional EER metric and additionally study the temporal distribution of errors in order to get insights into the robot's ability to be continuously authenticated by the system.

The rest of the paper is organized as follows: In Section 2 we discuss related work. In Section 3 we describe our data collection experiments and feature extraction approach. In Section 4 we discuss the design of the attack and present our attack performance evaluation in Section 5. We finally make our conclusions in Section 6.

2. Related Work

2.1. Robotic Attacks

The most related paper to this study is our work in [15]. In that paper, we showed a Lego robot to execute touch gestures that increased the EER of the touch-based authentication system relative to when humans attacked the system. While that work demonstrated that a robotic device could execute touch strokes on the phone, there are several reasons as to why the Lego attack would not suffice as the robotic attack model that could be adopted for the burgeoning domain of touch-based authentication. Below, we discuss some of these reasons:

Ease of attack adaptation to different devices: A Lego robot basically consists of a set of pieces that are combined together into a structure that can be programmed to undertake a given task. In this case (i.e., in [15]), the pieces were combined to execute touch strokes on a phone placed horizontally on the ground. If one were to launch a similar attack on a laptop that has a touch screen, they would have to destroy the robot's structure, rebuild it to match the physical orientation and dimensions of the new device (e.g., a laptop would have a vertical or close-to-vertical screen which is much wider than that of a phone) and finally write new code that can run the motors in accordance with the new physical configuration. For our autonomous humanoid robot (i.e., the NAO H25 Evolution [5]) on the other hand, all these changes can be made by only tweaking the code since its mechanical framework is already shaped similarly to a human.

Precision of gesture execution: The attack in [15] executed zig-zag strokes due to the limited capabilities of the Lego. The Android system was "fooled" to "believe" that the zig-zag strokes were near-smooth curves (like those of a human) because it only sampled a subset of the points on the zig-zag path. Clearly if touch-based authentication got widely deployed, the default Android sampling rate would easily be increased to detect the true shape of the gestures. Our gesture execution mimics the human hand movement and thus executes gestures that are as precise as those of a human.

Ability to execute arbitrary gestures: By making trivial changes to our algorithms, our attack can be set to execute touch gestures with different properties (e.g., shapes, dimensions, etc.). This again naturally follows from a humanoid robot having joints that are designed to very closely match those of a human. For a Lego robot to execute gestures with shapes different from those in [15], a completely new physical structure would have to be built.

Attack performance evaluation: Aspects of the performance evaluation method in [15] made it challenging to asses the full impact of the robotic attack presented in [15] (details of this challenge already discussed in Section 1).

These factors, among others cement the need for a more rigorous robotic impostor testing approach for touch-based authentication systems, which in turn motivates this work.

2.2. Touch-based Authentication

The other stream of related papers to our study is that of works which developed mechanisms that continuously authenticate users based on the gestures they execute on the touch screen (e.g., see [9][14][8]). One thing that is common between all these works is that performance evaluation

is based on the assumption of impostors who make no effort to forge a given swiping pattern (i.e., zero-effort impostors), which in turn is the major factor that puts these works apart from our paper. We do not discuss details of the authentication techniques used in these papers due to space limitations.

3. Data Collection and Feature Extraction

After getting IRB approval from out University, we collected data from 33 users. The data collection experiment involved users responding to a series of multiple choice questions on the smart phone. Answers to these questions were available on the same page on which the questions were presented, however, users had to scroll back and forth to find these answers before entering them. This back and forth swiping generated the touch strokes (or swipes) that we used for our authentication investigations. Each user participated in two different data collection exercises on two different days, with data from one day used for classifier training and data from the other day used for testing.

To extract features from a user's touch strokes, we used the sliding window mechanism like has been done in past work (e.g., see [9][14]). Each window comprised 10 strokes from which 28 features were extracted per stroke. The features were computed as follows: From each of the velocity, pressure, area (between finger and screen) and acceleration measurements at different points along a stroke, we calculated the first, second and third quartile and the mean and standard deviation, giving a total of 20 features (i.e., 5 statistical measures $\times$ 4 categories of readings). To these 20 features we added the start and end coordinates of each stroke (a total of 4 features since each X and Y value is considered a separate feature) and the following four features: (1) the angle between the end-to-end line joining the ends of a stroke and the horizontal, (2) the length of the end-to-end line, (3) the sum of distances between consecutive touch points, and, (4) total time to execute a stroke. From the ten strokes comprising a sliding window, component-wise means of the ten 28-dimensional feature vectors were computed to produce a single 28-dimensional vector. Note that with the sliding window mechanism, a user who executed n strokes had $n - 9$ feature vectors in total.

4. Attack Design

4.1. Threat Model

Our interest is the question of whether or how a humanoid robot could be used to execute touch strokes that match those of a human, also referred to as the victim. Our attack design thus assumes the scenario of an adversary who gets access to a user's swiping samples (or template), and then uses this information to train a robot to mimic the touch strokes. This assumption of an attacker who steals biometric samples and later uses them as input to design and launch attacks against the victim is quite standard in threat models used for the performance evaluation of behavioral biometrics installations (e.g., see [10]).

4.1.1 Threat Examples

With a motive to steal the victim's phone at a later time, the attacker could in practice get access to the intended victim's swiping data through social engineering, i.e., tricking the unsuspecting victim to swipe on the attacker's phone which has malware installed to record the victim's touch patterns. This trickery could for instance be done by asking the (intended) victim to browse some images or read some text on the attacker's phone.

The other way in which swiping samples could be stolen is where the victim's phone itself has malware that logs information and sends it to servers controlled by the adversary (e.g., malware similar to that reported on Sony Xperia phones recently [6]). Samples stolen in this way from high value targets could then later be marketed in underground hackers' networks. Like was assumed in the earlier cited Lego attack [15], the attacker's aim in both cases will be to read sensitive private information on a victim's phone whose only active layer of defense is the touch-based authentication system.

There is no doubt that the manures described above would require significant commitment from the adversary. Regardless of the attack implementation challenges however, the question of how or whether a sophisticated adversary (e.g., one using a robot) can mimic a given user's swiping pattern to defeat authentication mechanisms remains very critical for the security evaluation of the technology. Our attack tackles this problem by providing a blueprint for the emulation of robotic adversaries against touch based authentication.

4.2. Learning a User's Swiping Behavior

During a swiping session, a user may execute strokes having a wide range of properties (e.g., with regard to length, speed, etc.) depending on the exact task being undertaken at the point in time. During the attack, the robot will need to be aware of these stroke categorizations so that it can generate a mix of strokes that map to all stroke families manifested in a user's swiping behavior. We address this challenge as follows: First, we use clustering to discover the different stroke families in a user's swiping data. From data stolen from the victim, we first created feature vectors from each stroke (i.e., approach to feature vector computation was described in Section 3), and then run the X-means algorithm [11] to categorize the feature vectors into clusters. The advantage of X-means over other partition-based algorithms such as K-means is that the num-

ber of clusters (which we don't know) is not required as an input. The method estimates the number of clusters by finding the cluster composition (i.e., numbers of clusters and centers of clusters) that optimizes the Bayesian Information Criterion. We used the default settings in R [12] to implement the algorithm on our data.

For a good number of users, the clustering process generated large numbers of clusters, some of which only contained very few strokes. We only retained about 60% of the clusters, ignoring the very small clusters as noise. As a hypothetical example, assume a user for whom we retained 3 clusters whose sizes were in the ratio 1:2:3. The strokes to be executed by our robot to attack this user are such that representative strokes from each of the clusters get executed while keeping the ratio of cluster sizes in mind. For each stroke that is to be executed, the robot probabilistically selects a cluster (probability determined by the above mentioned ratio) and then creates a stroke matching the average properties of the strokes in the cluster. The average properties referred to here are the mean values of the velocity and the start and end coordinates of the strokes within a cluster. Other properties such as pressure and area are controlled through the nature and placement of the stylus being used to swipe (details in Section 4.3).

Given a cluster for which the mean value of velocity across all strokes is v, each of the strokes forged for this cluster will have a velocity $v + \beta.\sigma$, where β is a random constant between -0.5 and 0.5 and σ is the standard deviation of the velocities of the strokes in the cluster. The same approach is used for all the other features, with the random constant being used to ensure that two strokes from the same cluster are not exactly the same.

4.3. Executing a Robotic Touch Gesture

Given the mean properties of a stroke as described above, Algorithm 1 illustrates how the robot executes the stroke (which is represented as the input argument *R*). The first step is to convert the pixel coordinates of the end points and start points of the stroke into what we term as "box locations" on the phone screen. This is done in steps #2 and #3 where $R[InitX]$ and $R[FinalX]$ are the initial and final pixel X coordinates on the phone (Y coordinates defined similarly), and X_i and X_f are the so called box locations mapping to these coordinates. Figure 1 illustrates our notion of box locations. The figure is a phone screen on which a grid has been superimposed. Because the robot finger (which is actually a stylus — see Figure 2) cannot precisely pinpoint an individual pixel value, we use the grid (or boxes) to group together a set of neighboring pixels. The robot's system thus sees the phone screen as if it is the grid mentioned above, and, given a pair of coordinates from a user's strokes, it interprets this input as the box location mapping to those pixel coordinates.

ALGORITHM 1: How NAO robot executes a touch stroke on phone screen

1 **Function** `MakeSwipe(`R`)`
2 $(X_i,Y_i) \leftarrow$ `GetInitBoxCoords(`***R[InitX], R[InitY]***`);`
3 $(X_f,Y_f) \leftarrow$ `GetFinalBoxCoords(`***R[FinalX], R[Final]***`);`
4 ;
5 $Angles_i \leftarrow (RSP_{Angle}[X_i][Y_i], RSR_{Angle}[X_i][Y_i], RER_{Angle}[X_i][Y_i], RWY_{Angle}[X_i][Y_i], REY_{Angle}[X_i][Y_i])$;
6 ;
7 $Angles_f \leftarrow (RSP_{Angle}[X_f][Y_f], RSR_{Angle}[X_f][Y_f], RER_{Angle}[X_f][Y_f], RWY_{Angle}[X_f][Y_f], REY_{Angle}[X_f][Y_f])$;
8 ;
9 `MoveHandAbove(`***InitAngles***`);`
10 `Swipe(`$Angles_i$, $Angles_f$, ***Velocity***`);`
11 `MoveHandAbove(`***FinalAngles***`);`
12 **end**

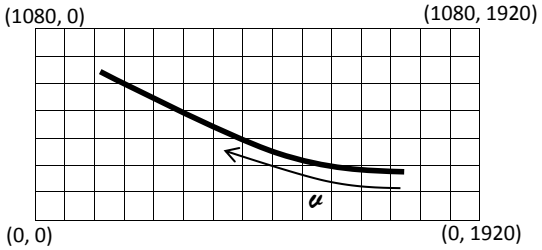

Figure 1. The NAO robot's view of the phone screen. The robot sees the phone as if it is an 18 × 9 grid (not shown to exact dimensions here), and interprets each pixel coordinate in terms of the cell to which it belongs. An example of a robotic touch stroke is shown on the grid. v is the mean velocity along the stroke.

To create a touch stroke between two box locations corresponding to the pixel coordinates at the ends of a stroke, we programmed 5 different motors in the robot's hand. Figure 3 shows the locations and directions of movement of these motors. The motors at the extreme right control the Right Shoulder Roll (RSR) and Right Shoulder Pitch (RSP) while the motors in the middle control the Right Elbow Roll (RER) and Right Elbow Yaw (REY). At the wrist we only controlled the Right Wrist Yaw (RWY). For more details on the motor specifications, the reader is referred to [4].

In a preliminary set of calibration experiments, we found the 5 motor angles corresponding to each box location (Figure 1) and the motor speeds needed to produce a range of swiping velocities on the screen. During the attack therefore, the robot did the reverse process — i.e., it mapped the

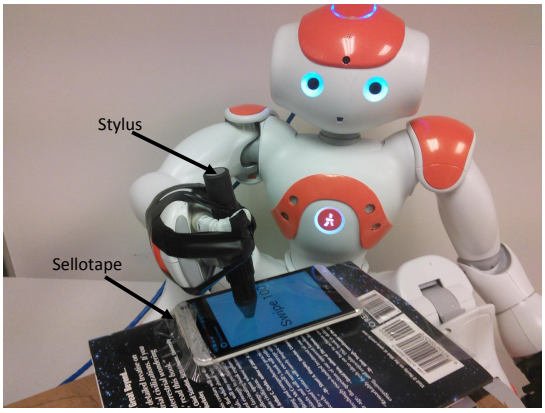

Figure 2. NAO robot swiping on the phone. The sellotape holds the phone in position during swiping. Also, the stylus is firmly held in the robot's fingers with the aid of sellotape.

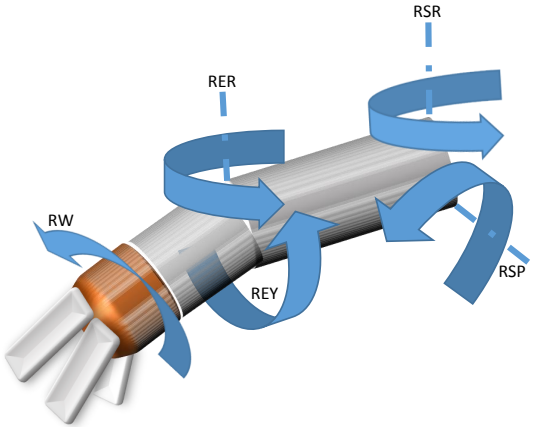

Figure 3. Illustrating the directions of rotation of the five motors that were programmed to create a touch stroke

swipe velocities (extracted from a user's data) back to motor speeds and the box locations (generated in lines #2 and #3 of Algorithm 1) back to motor angle values, enabling it to navigate to the required positions on the grid at the required velocity. Lines #5 and #7 of Algorithm 1 respectively show how the robot maps the start and end points of a stroke to the motor angles ($Angles_i$ and $Angles_f$) needed to perform the required motion. In line #9 the robot moves the stylus to the initial position just above the phone screen, and executes the swipe action between the initial and final angles in line # 10. In line #11 the stylus is moved off the phone, ready for execution of the next touch stroke.

We used a cosmonaut stylus as it registered the required touch area of about 0.15 units. We (programmatically) located the robot's hand (or stylus) relative to the phone in such a way to get a pressure that ranged between 0.4 and 0.6 units during swiping. These configurations were again set based on a preliminary set of calibration experiments. In [15] we used the same values of area and pressure since they were representative of the average behavior across the population. Between the two end points of a stroke, the intermediate motor angles were set to cause each stroke to have a very slight curvature (i.e., deviating very slightly from the straight line) since this was the commonly observed behavior across the population.

5. Performance Evaluation

In this section we evaluate how the robot performed at forging/copying users' touch gestures. We also compare the performance of the robotic attack with that of a zero-effort attack (described in Section 2.2) so as to give insights into how our attack compares with the state-of-the-art performance evaluation methods (which are centered around the zero-effort attack).

5.1. Training and Testing Process

For each user, data from one session was used for classifier training while data from a second session was used for testing (review data collection Sessions in Section 3). Each user's training template was built based on 80 touch strokes. Fixing this strokes requirement helped us prevent biases in the classification performance that might have arisen out of users having varying extents of training. Since only 26 of the original set of 33 users met this 80 strokes requirement in the testing set, we focused our performance evaluations on this sub-set of users. To compare the impact of the robotic attack with that of attacks launched by humans, we carried out two types of impostor attacks : a robotic attack in which 100 samples generated by the robot were used to attack each user's template, and a human impostor attack in which 100 samples were drawn randomly from users other than the particular user under investigation.

For each form of impostor attack we also carried out the so called "genuine attack" (i.e., the user's own samples being used to attack the user's template), so as to be able to generate a Detection-Error Trade-off (DET) curve for each user. The genuine attack was also based on 100 samples from the genuine user. In the following subsection we compare the mean Equal Error Rates (EERs) obtained when human impostors were used to those obtained when robotic impostors were used. Classification was done based on 3 well-known verifiers, namely the Support Vector Machine (SVM), Multilayer perceptron (MLP) and the K-Nearest Neighbors classifier (K-NN). We used k=5 for the latter classifier and default settings in R [12] for all other classifiers.

5.2. Mean Error Rates

Figure 1 shows the mean EERs across the population obtained under the robotic attack compared side-by-side with those obtained under the human attack. For all three classifiers, observe that the mean EER (μ_{EER}) under the robotic attack was higher than that under the human attack. The

Table 1. Comparing the mean EERs obtained when the robotic attack was launched to those obtained when human impostors attacked the system

Classifier	Human Attack		Robotic Attack		% Increase	
	μ_{EER}	σ_{EER}	μ_{EER}	σ_{EER}	μ_{EER}	σ_{EER}
SVM	0.081	0.112	0.363	0.273	348.0	144.4
MLP	0.101	0.123	0.268	0.207	165.8	67.8
KNN	0.167	0.183	0.322	0.265	92.9	44.8

same applies to the standard deviation (σ_{EER}) of the EERs which was higher under the robotic attack than under the human attack. The last two columns express these increments as percentages, showing that μ_{EER} and σ_{EER} increased by up to 348% and 144% in the extreme case. In general, the heightened EERs indicate that the system saw more cases of impostor access during the robotic attack, while the high standard deviations indicate that the system became more unreliable (unpredictable) as certain users had much higher EERs than the rest of the users.

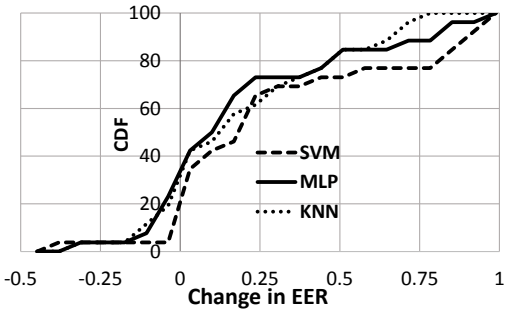

Figure 4. CDF of the change in EER per user due to the attack.

Figure 4 gives some insights into the impact of the attack on each user. For each user we subtracted the EER obtained during the zero-effort attack from that obtained during the robotic attack and plotted the CDF of these differences. A negative value of this difference indicates that the user had a lower EER under the robotic attack (and thus had a template more resistant to the robotic attack than the zero-effort attack) while a positive value indicates that the user in question was more susceptible to the robotic attack than the zero-effort attack. Figure 4 shows that across the 3 verification algorithms, between 80% and 65% of the population performed worse under the robotic attack (or had positive differences). A possible reason as to why some users had lower EERs under the robotic attack is that some of the forged variables that were based on population statistics and not each user's own behavior (e.g., see pressure and area in Section 4.3) may not have matched well with certain users' templates. In general however, Figure 4 shows that the robotic attack achieves great success at forging users' swiping behavior.

5.3. Temporal Distribution of False Acceptances

As already argued in Sections 1 and 2, the EER-based evaluation presented in the previous section portrays an incomplete picture of the impact of the attacks because it does not provide any information on when classification errors occur. In this section we compare information on when classification errors occurred during the robotic and human attacks. We particularly focus on *false acceptance* errors, since we are interested in studying the success of the attack (i.e., the cases when the system misclassified the attack samples as legitimate samples). This comparison should give some measure of the likelihood that the attacker (human or robotic) can *continuously* authenticate on the system.

Assume an attacker who makes n authentication attempts on a swiping-based authentication system and successfully authenticates (i.e., registers a false acceptance) during the i^{th} attempt. If the next successful authentication attempt during this swiping session is the j^{th} attempt, then we define $D_F = j - i$ as the distance between adjacent false acceptances. If D_F =1, this means that two consecutive authentication attempts made by the attacker were successful. If D_F is large, this means a good number of attempts by the attacker are rejected before a given attempt gets falsely accepted.

In general, the lower the value of D_F, the higher the likelihood that the attacker will successfully continuously authenticate against the user's biometric profile. If D_F is large, there is a good chance that an alarm will be triggered during the many failed authentication attempts registered in close proximity to each other. For each user and each form of attack, we compute the mean value of D_F (denoted as $\bar{D}_F$) to get some indication of the attacker's likelihood to continuously authenticate on the system. Table 2 summarizes the findings from this analysis. We bin $\bar{D}_F$ in buckets of size 2 and count the percentage of users falling in each bin for each type of attack and verification algorithm (i.e., each of the six right-most columns in the table add up to 100%).

Take the case of $0 < \bar{D}_F \leq 2$ for instance. For all verification algorithms, Table 2 shows that 0% of the users had $\bar{D}_F$ in this range for the human attack. This is in comparison to between 19.2% and 34.6% of the users for the worst and best performing verifiers respectively. A similar trend is seen with the interval $2 < \bar{D}_F \leq 4$, where again larger percentages of users are seen for the robotic attack than for the human attack for all verification algorithms. For higher values of $\bar{D}_F$ a reverse trend is seen, as more users are seen under the human attack than under the robotic attack. These results confirm that the robotic attack had a much higher chance of maintaining a sustained continuous authentication session than the zero-effort impostor attack.

Table 2. Percentage of users for each range of $\bar{D}_F$ for different classifiers and impostor attack types.

$\bar{D}_F$	KNN		MLP		SVM	
	Human	Robot	Human	Robot	Human	Robot
$0< \bar{D}_F \leq 2$	0.0%	34.6%	0.0%	19.2%	0.0%	23.1%
$2< \bar{D}_F \leq 4$	3.8%	30.8%	3.8%	19.2%	0.0%	19.2%
$4< \bar{D}_F \leq 6$	11.5%	3.8%	7.7%	3.8%	11.5%	11.5%
$6< \bar{D}_F \leq 8$	15.4%	3.8%	7.7%	11.5%	7.7%	11.5%
>8	69.2%	26.9%	80.7%	46.2%	80.7%	34.6%

6. Conclusions

In this paper we have presented an attack in which a humanoid robot mimics the touch gestures executed by humans. Using a combination of the traditional error metrics and a distance measure that captures the robot's likelihood to continuously authenticate, we have showed the robotic attack to achieve significant success at forging users' touch gestures. The paper does not only highlight the threat that robots pose to touch-based authentication systems, but also takes steps towards presenting a general robotic attack framework which simulates sophisticated adversaries that touch-based authentication systems may face in practice.

7. Acknowledgment

This work was supported by DARPA Active Authentication grant FA8750-13-2-0274.

References

[1] Darpa-baa-13-16 active authentication (aa) phase 2. https://www.fbo.gov/index?s=opportunity&mode=form&id=aa99ff477192956bd706165bda4ff7c4&tab=core&_cview=1. Last accessed in April, 2013.

[2] Dhs s and t cyber security division 5-year broad agency announcement. https://www.fbo.gov/index?s=opportunity&mode=form&id=0e4bc523b788f020ab44ce8cd24a8d41&tab=core&_cview=1. Last accessed in Jan, 2015.

[3] Innovative cross-domain cyber reactive information sharing (iccyris). https://www.fbo.gov/index?s=opportunity&mode=form&id=51735e41343a6e5ee5014b3a6c8bde3f&tab=core&_cview=1. Last accessed in Jan, 2015.

[4] Nao software 1.14.5 documentation. http://doc.aldebaran.com/1-14/family/nao_h25/motors_h25.html. Last accessed in March, 2015.

[5] Robots lab. http://www.robotslab.com/#gsc.tab=0. Last accessed in Jan, 2015.

[6] Security affairs. http://securityaffairs.co/wordpress/29703/malware/sony-xperia-china-spyware.html. Last accessed in March, 2015.

[7] P. Bours. Continuous keystroke dynamics: A different perspective towards biometric evaluation. *Inf. Secur. Tech. Rep.*, 17(1-2):36–43, Feb. 2012.

[8] S. Govindarajan, P. Gasti, and K. Balagani. Secure privacy-preserving protocols for outsourcing continuous authentication of smartphone users with touch data. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–8, Sept 2013.

[9] F. Mario, B. Ralf, M. Eugene, M. Ivan, and S. Dawn. 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, 2013.

[10] T. C. Meng, P. Gupta, and D. Gao. I can be you: Questioning the use of keystroke dynamics as a biometric. In *NDSS,2013*, Feb 2013.

[11] D. Pelleg and A. W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 727–734, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[12] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[13] N. Sae-Bae, K. Ahmed, K. Isbister, and N. Memon. Biometric-rich gestures: A novel approach to authentication on multi-touch devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 977–986, New York, NY, USA, 2012. ACM.

[14] A. Serwadda, V. Phoha, and Z. Wang. Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–8, Sept 2013.

[15] A. Serwadda and V. V. Phoha. When kids' toys breach mobile phone security. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, CCS '13, pages 599–610, New York, NY, USA, 2013.

[16] A. Serwadda, Z. Wang, P. Koch, S. Govindarajan, R. Pokala, A. Goodkind, D.-G. Brizan, A. Rosenberg, V. Phoha, and K. Balagani. Scan-based evaluation of continuous keystroke authentication systems. *IT Professional*, 15(4):20–23, July 2013.