K-means++ vs. Behavioral Biometrics: One Loop to Rule Them All

Parimarjan Negi *, Prafull Sharma †, Vivek Sanjay Jain ‡ and Bahman Bahmani §
Stanford University
Email: *pnegi@stanford.edu, †prafull@cs.stanford.edu, ‡vsjain@alumni.stanford.edu, §bahman@cs.stanford.edu

***Abstract*—Behavioral biometrics, a field that studies patterns in an individual's unique behavior, has been researched actively as a means of authentication for decades. Recently, it has even been adopted in many real world scenarios. In this paper, we study keystroke dynamics, the most researched of such behavioral biometrics, from the perspective of an adversary. We designed two adversarial agents with a standard accuracy convenience trade-off: Targeted K-means++, which is an expensive, but extremely effective adversarial agent, and Indiscriminate K-means++, which is slightly less powerful, but adds no overhead cost to the attacker. With Targeted K-means++ we could compromise the security of 40-70% of users within ten tries. In contrast, with Indiscriminate K-means++, the security of 30-50% of users was compromised. Therefore, we conclude that while keystroke dynamics has potential, it is not ready for security critical applications yet. Future keystroke dynamics research should use such adversaries to benchmark the performance of the detection algorithms, and design better algorithms to foil these. Finally, we show that the K-means++ adversarial agent generalizes well to even other types of behavioral biometrics data by applying it on a dataset of touchscreen swipes.**

I. Introduction

Behavioral biometrics is the study of an individual's unique behavioral patterns, such as hand-writing, typing, or mouse movements. It has been researched as a source of recognition, and authentication, for a long time. For instance, handwriting recognition began to be scientifically studied in the early twentieth century [31], and keying patterns of telegraph operators during World War II were used to identify them [36]. In its modern incarnation, researchers have studied such techniques in the context of keystrokes, mouse movements, smartphone swipes, gait analysis and so on. Combined with an increased access to behavioral information using sensors on mobile phones and other electronic devices, behavioral biometrics has also seen a surge in real-world adoption.

Many banks already use typing information as an additional layer of security [30], startups provide APIs for password hardening [9], or even password-less logins [23], and Google is developing methods to authenticate users on mobile devices without passwords [20]. These behavioral biometrics such as keystroke dynamics and touchscreen swipes are used as a secondary authentication method along with primary authentication methods, such as password correctness.

The need for such secondary authentication methods have become apparent with increasing awareness that human chosen passwords are far from safe, [8], [10], [13], [16], [22]. It is common practice to use additional layers of security, using methods such as two factor authentication [17], but such explicit methods are usually disruptive to the user. Behavioral biometrics has the promise of being a second layer of authentication that can be seamlessly integrated into the current authentication systems - for instance, in keystroke dynamics, the additional security layer would be based on the key-press timings as the user enters his password. The authentication system can then do an additional check in the background without affecting the user experience.

Since keystroke dynamics is the most studied of the behavioral biometric approaches, we focus on designing our adversarial algorithms in that context. Moreover, we show that such an adversarial agent can also be applied to a different dataset involving touchscreen swipes.

We focus on a scenario where the attacker has access to the user's password, but needs to overcome a keystroke dynamics based authentication layer. This can happen when the authentication information of a website is leaked, and the attacker gains access to the passwords for that website [8], [10], [13], [22]. Then, the attacker would want to test whether the same username and password is used on a sensitive (e.g., banking) website, but the bank may be utilizing keystroke dynamics in its authentication system. Since, the information for the keystroke dynamics models will probably be stored by a different company, whose API the bank uses, such a data breach is unlikely to compromise this information. The naive approach for the adversary would be to manually type in the password, or use an automated tool to send the key-presses. Keystroke dynamics systems have been shown to be extremely robust to such adversarial attempts.

Such a system can be bypassed if the attacker can access the user's typing data. This is possible using social engineering techniques like luring the victim to a web-page, but it is another attack model with different trade-offs, e.g., it reduces the attack coverage considerably, as many users would avoid falling for it.

In our attack model, the attacker has no other information about the user besides the username and password. Thus, the

Research supported in part by Stanford Cyber Initiative and The Hewlett Foundation

Network and Distributed Systems Security (NDSS) Symposium 2018
18-21 February 2018, San Diego, CA, USA
ISBN 1-891562-49-5
http://dx.doi.org/10.14722/ndss.2018.23303
www.ndss-symposium.org

objective of the attacker is to bypass the authentication system in as few tries as possible. In practice, the attacker could spread his attempts over several weeks in order to bypass the website's restrictions, e.g., a limited number of login tries.

In this domain, past research has focused on a mode of attack in which the attacker collects a large number of samples of other users typing the same password, and utilize this data to generate adversarial samples. In this scenario, we improved upon the best known methods to generate samples by designing an adversary, Targeted K-means++, which searches through the space of potential samples much more efficiently. This represents a serious vulnerability if such systems are used to protect sensitive accounts. But it does justify the use of behavioral biometrics as it requires the adversary to spend additional resources in collecting samples from other people. Therefore, next we designed a novel adversary, Indiscriminate K-means++, that generates the adversarial samples from pre-computed data from the general population. This diminishes the value of behavioral biometrics as it does not impose any additional costs on the attacker.

Along the way, we also collected a large dataset using Amazon Mechanical Turk to test various hypotheses [1]. We found that our adversarial agents could bypass state of the art classifiers within a few tries for a majority of the users.

The paper is structured as follows. In section two, we provide an overview of the past behavioral biometrics research and adversarial models that inspired ours. In section three, we describe the datasets, and our adversaries along with the intuition behind the attacks and analysis of their algorithms. In section four, we present our experimental setup, which includes a description of the protocols used, and the experimental results. We conclude the paper in section five with a discussion about our contributions and the future work that could benefit the field of behavioral biometrics.

II. Related Work

A. Behavioral Biometrics

Behavioral biometrics rely on patterns in user interactions with input devices. Traditionally, these included keyboards, [11], [15], [19], and mouse [2], [18], [38]. Now, modern smartphones also provide an array of sensor information, which can be used similarly to construct user profiles based on touchscreen swipes [4], [14], gait analysis [12], [27], and other metrics. These signals are collected from the legitimate user and then analyzed at authentication time to verify the identity of the user attempting to log in.

One of the landmark papers comparing many of the state of the art algorithms in keystroke dynamics is [19]. In this paper, the author published the results of different classification algorithms, along with the benchmark DSN dataset. Since many subsequent papers in the field have experimented based on these results, we train many of the same classifiers, and present our results on the benchmark DSN dataset as well.

The current state of the art for the classifiers is represented by the Keystrokes Biometrices Ongoing Competition (KBOC) [26]. Many new techniques to improve the performance of keystroke dynamics classifiers were developed there. In particular, we utilized the classifiers developed by the winner of the competition, Vinnie Monaco [24], [25], along with some of his key algorithmic techniques.

In comparison to keystroke dynamics, touchscreen swipes have not been studied much as a source of authentication. Antal et al. [4] recently published a comprehensive analysis of many classifiers, along with a standard dataset that had features extracted from swipes on an Android platform. In this paper, we closely follow the experimental setup and classifiers used in [4] in order to train the classifiers based on touchscreen swipes.

B. Adversarial Machine Learning

The field of adversarial machine learning studies attacks against machine learning algorithms. Machine learning methods have been designed by assuming various properties about the underlying data (e.g., linear separability), but an adversarial player may not necessarily abide by such assumptions, and can actively attempt to foil the model. Adversarial algorithms against machine learning based systems can be categorized into a taxonomy based on three aspects:

1) **Influence:** Causative vs. Exploratory: This determines if the attack is performed at training time (Causative) or test time (Exploratory).
2) **Security violation:** Integrity vs. Availability: This determines whether the attack is aimed to allow an attacker to bypass the system (Integrity) or to block a legitimate user from accessing the system (Availability).
3) **Specificity:** Targeted vs. Indiscriminate: This determines whether the attack is aimed at a particular data point such as a particular user (Targeted), or is aimed broadly at a population (Indiscriminate).

For further details on this taxonomy, please refer to the overview paper by Barreno et al. [6]. The adversarial agents we designed were inspired by the broader context of this framework.

C. Adversarial biometrics

Even though there has been significant amount of research on behavioral biometrics, in particular keystroke dynamics, very little research has focused on generating adversarial samples to bypass such systems. A few papers [33], [37] have studied adversarial attacks on keystroke dynamics assuming they had access to varying amounts of the target user's typing data. As mentioned in the introduction, this is a different attack scenario than the one studied in this paper. Also, in the most common attack scenarios, such as when passwords are leaked, it is unreasonable to expect the attacker to have any information about the target user besides the username and password.

Stefan et al. [35] and Serwadda et al. [1], [34] both studied similar attack scenarios to this paper. Both these papers generated their adversarial samples using impostor data from other users typing the same password as the target user. This is the same scenario as the Targeted K-means++ adversary designed in this paper. Stefan et al. used an adversary which attempted

[1] https://github.com/parimarjan/adversarial_keystrokes/tree/master/datasets

to model the key-press timings as a Gaussian distribution. They concluded that the classifiers were robust to such adversarial samples. But it is difficult to draw any conclusions from this because they used very little data (around 20 users, and 35 samples per user for each password). They also provided a systems level interface to record keystroke times of typing patterns, and a way to inject adversarial timings into this system. In contrast, we assume an attack scenario based on an online model. This leads to differences in the data collection phase. We also don't focus on the process of injecting the attacks into the web browser as this is a straightforward task using automation tools.

Serwadda et al. [1] used a large dataset, which is not publicly available, to study the same scenario. They designed an adversarial agent called MasterKey. It generated the adversarial samples by starting from the mean of impostor samples, and perturbing values to explore the rest of the sample space of possible key-press timing values. They found that just with a single guess, an attacker is able to compromise the security of approximately $5-30\%$ of users. However, even after hundreds of guesses, their solution could not break into a majority of the users. In order to compare our algorithm to MasterKey, we implemented a version of this attack as well. In the DSN dataset, it actually performed even better than in the original study. In Serwadda et al. [34], they further expanded on their work in [1]. In particular, they did extensive statistical analysis of individual keystroke features. There conclusion that the features follow independent probability distributions is utilized by us when designing the Indiscriminate K-means++ adversary.

III. Datasets and Methodology

In this section, we present our adversarial attacks, and provide the intuition behind their effectiveness. These details require an understanding of the datasets, and features, involved, so we start by presenting those in detail. Our extensive experiments, presented later in this paper, empirically verify the effectiveness of our attacks.

1) Datasets: We performed experiments with two existing datasets, one on keystroke dynamics and one on touchscreen swipes. We also collected a larger dataset of keystroke dynamics using Amazon Mechanical Turk. These datasets are described below.

DSN Dataset: For the first set of experiments we used the dataset by Killourhy and Maxion [19], which consists of 51 users, each typing the same imposed password, ".tieRoanl", 400 times each. DSN is a widely established dataset and numerous papers have explored algorithms on it, thus making it easy to compare different approaches and verify our own results.

MTurk Dataset: We collected data of nearly 600 users typing five common passwords (presented in Table I) 100 times each. These passwords were chosen from a list of most common passwords. Thus they are representative of the typical passwords in the wild. A few of the motivating reasons behind collecting this dataset were:

- The DSN dataset used an uncommon and difficult to type password, ".tie5Roanl". It is possible that the users never quite got used to it, hence their typing patterns weren't sufficiently unique. In the MTurk dataset the passwords are common English words, so the users should be used to them from the start.
- We wanted to ensure that our results, particularly for the adversarial attacks, translate to real world scenarios where the data would typically be collected on the Internet.
- We wanted to replicate the results across a much larger sample pool, and across different passwords.

Touchscreen swipes dataset: This touchscreen swipes dataset was collected by Antal et al. [4] using a psychological personality based questionnaire on an Android smartphone. The published paper has their analysis with various classifiers for 40 users, but since then, they have published a bigger dataset with 98 users [3], which is what we used in our experiments.

2) Features: For both the keystroke datasets mentioned above, we used the following three features, proposed by Killourhy-Maxion [19]:

- Press-Release: Duration a key was held down for.
- Release-Press: Delay between releasing a key, and pressing the next one.
- Press-Press: Delay between pressing the first key and the next key. This is just the sum of the previous two. It is not particularly necessary to use this feature - but since it was used in the paper by Maxion et al. [19], we decided to stick with the convention.

For designing the Indiscriminate K-means++ adversary we just used the first two features, and derived the third feature by taking their sum.

The touchscreen swipes dataset provided the following set of 11 features:

- duration: time between touch down and touch release
- length of trajectory: the length of the segment defined by the two endpoints
- average velocity: a fraction of the length of trajectory and duration
- acceleration start: the average acceleration at the first 4 touch points
- mid-stroke pressure: the pressure at the middle point of the swipe
- mid-stroke finger area: the finger area at the middle point of the swipe
- mean pressure: average of pressures in touch points
- mean finger area: average of finger areas in touch points
- gravity (x-axis): average of x gravities in touch points
- gravity (y-axis): average of y gravities in touch points
- gravity (z-axis): average of z gravities in touch points

More information about the experimental setup for this dataset can be obtained in [4].

A. Evaluating Classifiers

Equal Error Rate: An ideal authentication system would always accept a genuine sample, and reject an impostor sample. In practice this is rarely the case. Thus, there are two kinds of possible errors: rejecting a correct input, or the False Reject Rate, (FRR), and accepting a wrong input, or the False Accept Rate (FAR). To measure the performance of a system, the keystroke dynamics literature focuses on Equal Error Rate (EER) which is the error when the acceptance threshold of a classifier is set to the value at which the FAR is equal to the FRR. EER is a value in the range $[0, 1]$ and a lower EER implies a lower error, and a better classifier.

In practice, setting the acceptance threshold of a classifier leads to an important trade-off between security (avoiding false acceptances) and usability (avoiding false rejections). Various keystroke dynamics startups provide the client the flexibility of choosing higher levels of security in their API calls. Therefore, to evaluate the effectiveness of our adversaries in this setting, we also used thresholds of differing "strictness" which models such a scenario.

B. Attack Intuition

At its extreme, the idea behind behavioral biometric authentication is that every person has unique patterns of behavior. On the other extreme, we could consider the claim that everyone's behavioral patterns are the same. Clearly, the reality must lie somewhere in between these two extreme viewpoints. In the domain of keystroke dynamics, various classifiers have been shown to be robust in distinguishing among genuine and impostor samples. But our intuition suggests that there must be significant overlap between the typing samples of many users.

It is reasonable to imagine that each person's "unique" typing style is really part of a bigger family of similar user styles. Thus, we hypothesize that the set of keystroke timing patterns of all users are clustered into a limited, and relatively small number of clusters, where users with similar typing behaviors belong to the same cluster. Our idea is to mimic the target user's typing patterns by generating all such clusters using data collected from the general population.

We support this hypothesis by analyzing the keystrokes data from the DSN dataset. We analyze the data by running K-means with different values of K and analyzing the average distance of a sample to their closest cluster center. Figure 1 presents two plots, plot (a) represents the average distance of a sample to the centroid of its cluster and plot (b) shows the first derivative of the function represented by plot (a). In the presented figure, we can see that average distance remains relatively similar with the increase in the number of clusters. This is supported by Plot (b) as the derivative also flattens out at $k = 10$. This supports the hypothesis that most users don't belong to their own individual clusters.

C. Attack Model

Now, we anchor our attack model in the broader framework of the taxonomy of adversarial algorithms described in section IIB.

Evaluating Adversaries: In terms of influence, our attack model belongs to the "exploratory" category as it targets the classifier at test time. The security violation falls naturally under the "integrity" category, as the adversary attempts to bypass the security provided by the classification system. In terms of "specificity" - both targeted and indiscriminate model attack scenarios that an adversary would be interested in. We designed our two adversaries based on the distinctions between these two scenarios. More specifically, we focused on a type of probing attack, called the "Adversarial Classifier Reverse Engineering (ACRE)", which was introduced by Lowd et al. [21]. In this framework, given an attacker's cost function, the goal is to find a lowest attacker-cost instance that the classifier labels as negative (i.e., it passes through). Various other papers in the literature have used such a model [7], [28], [29]. Specifically, we consider how many tries does it take an adversary to compromise the security of a classifier.

D. Adversary I: Targeted K-means++

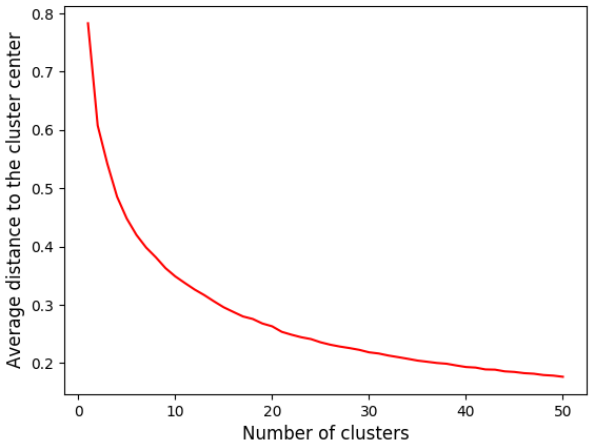
Here we assume that the attacker has access to a large pool of sample data (i.e., timings of many other users typing the target user's password), but does not have data from the user they wish to impersonate. This is a reasonable scenario: The attacker could get many user's keystroke timing information about a given password by simply asking people to type the desired password on a paid crowdsourcing platform like Amazon Mechanical Turk. This may be an expensive process - but if the authentication system protects sensitive information, this would not be a major obstacle for the attacker.

The aim of the adversary is to efficiently explore the samples from different users in order to find candidates from the "cluster" of the target user. The simple approach here would be to run K-means for a particular value of k and try all the centroids. This performs reasonably, but it does not give us any good way to choose a value of k. Also, since K-means is a local search algorithm, its clusters can change considerably for different values of k.

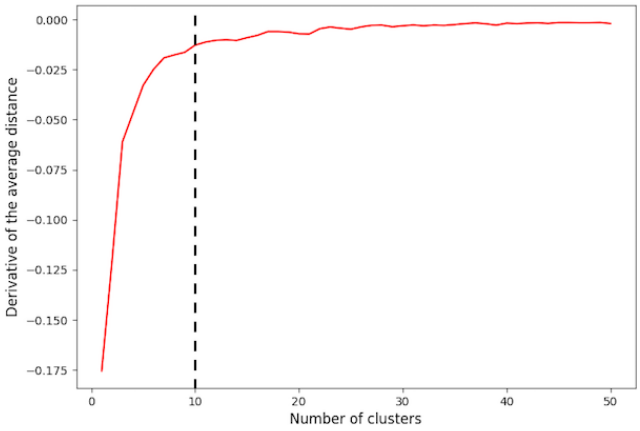
K-means++ is traditionally used as an initialization step for the centroids of K-means. It uses an iterative algorithm where the first center is selected at random from the data, and then each subsequent center is selected with a probability proportional to its contribution to the overall error given the previous selections. Intuitively, K-means++ exploits the fact that a good clustering is relatively spread out, thus when selecting a new cluster center, preference should be given to those further away from the previously selected centers. This fact is crucial for designing our adversary: if we fail to break the classifier's defenses with the try i, then the try $i+1$, would find a sample that is far away from the previous try - thus increasing the likelihood that it lands closer to the space of the target user's samples.

Theoretically, it has a nice property that when running it for $k+1$ iterations, the first k centers it generates are the same ones as it would have generated had it been run for k iterations in the first place. These initial set of centers is also provably close to the optimum solution [5]. As a result, using K-means to generate queries would require $O(K^2)$ queries to explore cluster counts 1 through k, but using K-means++ only requires k queries for the same goal.

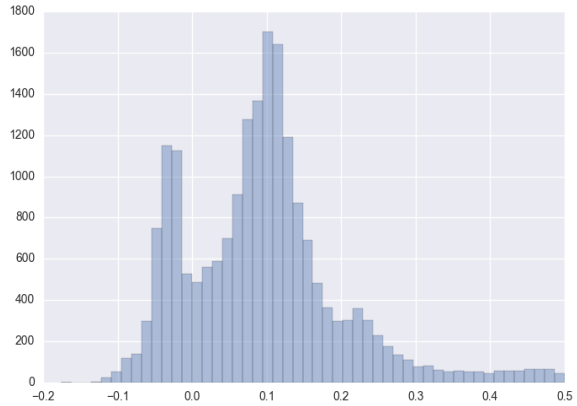
Based on the properties above, we repurposed the K-means++ algorithm, (see Algorithm 1), for this task. For the



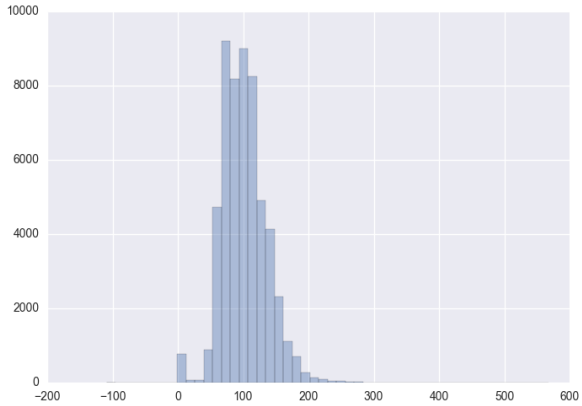
(a) We ran K-means on the DSN dataset for different values of k. Here we present average distances of a sample to their closest cluster center



(b) First derivative of Plot (a). This shows that the derivative flattens around $k = 10$. Thus, higher values of k do not improve the clustering by much.



(c) probability distribution of timing of digraph "an" while typing password ".tieRoanl" in the DSN dataset.



(d) probability distribution of hold time of "n" while typing password "letmein" in the MTurk dataset.

Fig. 1: Preliminary statistical analysis of keystrokes data

Algorithm 1 Adversarial Targeted K-means++

INITIALIZE $Try_1 \leftarrow$ the mean of the collected adversarial data set $\mathscr{X}$
INITIALIZE $Auth \leftarrow False$
INITIALIZE $i \leftarrow 2$
while $!Auth$ **do**
  $D(x) \leftarrow$ distance from nearest Try chosen so far to point x $(\forall x \in \mathscr{X})$
  $Try_i \leftarrow x \in \mathscr{X}$ with probability $\frac{D(x)^2}{\sum_{x' \in \mathscr{X}} D(x')^2}$
  $Auth \leftarrow$ True if Try_i passes the authentication
  $i++$
end while

first try, rather than choosing it randomly from the data, we select the mean of the given samples. Each new centroid selected by the algorithm comprises a new probe to the detection algorithm.

E. Adversary II: Indiscriminate K-means++

The main shortcoming of the targeted scenario is that collecting many samples for a single password is an expensive task. For instance, if the attacker is just trying out many passwords recovered from a leaked database, collecting samples for each one of them is highly impractical. So we design an adversary who may be willing to sacrifice some accuracy for convenience. In particular, the adversary may not be able to, or may not want to, collect a large sample of keystroke data for the target password. Instead, he may have access to a large pre-computed database of user's typing data. This scenario has never been studied before in the keystroke dynamics literature.

The key insight for designing this adversary is that given a target user's password, we can generate reasonable timing vectors if we have general population timing data for duration

of each key, and the time spent between successive key-presses. Every digraph, e.g., "as", and "at", would follow a different distribution. This could be due to several reasons, such as the distance between the keys. But there are only a limited number of such key-presses, and digraphs, possible - and it is not hard to imagine that data could be collected, and made publicly available, for all such cases. For instance, this could be done using botnets and keyloggers to get data from unsuspecting users.

The next challenge is generating the probability distribution for each key-press and digraph. One possibility is to directly sample from the empirical distribution (i.e., the collected data), hoping that if it is sufficiently large, it may represent the true population distribution. Instead, we chose to model it with a roughly correct distribution. Based on eyeballing the distributions (see Figure 1), we chose a Gaussian mixture model with two components. One major advantage of this approach is that it is the most practical - a database with thousands or millions of samples for all possible digraph pairs will become too large. But, in this scenario, the adversary would only need to know the parameters of the distribution. More crucially, we did not wish to get the best fitting distribution - instead we wanted a convenient way that can let us easily generate a lot of reasonable samples. Then, we use this distribution to generate the desired sample size of timing samples of the key-presses for the target password. This situation is identical to the scenario for the Targeted K-means++ adversary described above. Thus, for the final step, we use the K-means++ algorithm to find the most efficient probes for the classifier. The intuition here is that even if our distribution is only a rough approximation of the real distribution - we can still expect it to find good probes by just sampling efficiently from the whole space. This algorithm is summarized in Algorithm 2.

We tested this adversary on each of the passwords in the MTurk dataset. For every digraph in the target password, we collected data with further experiments in which new users typed different words that included those digraphs I. For each digraph in the target password, we had the user type in two words, ten times each, that included the same digraph. For instance, when targeting "mustang", we had two words that include "mu", two words that included "us", etc. These words were chosen randomly with the criterion that they should not have more than a two character subsequence in common with the target password (e.g.,"must" has four characters in common with mustang, so it was not used). In reality, if the adversary has access to a large database of typing data, it is likely that he may find timing data on words that have a more significant overlap with the target password - especially since so many passwords use dictionary words. This would clearly only benefit the adversary.

We tested this adversary only on the MTurk dataset as we could control the method of timing extraction - the impostor samples were collected using the same JavaScript code as the target users data in the MTurk dataset, while the DSN dataset was collected on a single computer, and we do not have access to the software used to collect it, which would likely lead to slightly different timing latencies.

Algorithm 2 Adversarial Indiscriminate K-means++

```
LET SAMPLE-SIZE be the desired sample size.
LET HOLD-TIME(c) ← A function that returns a sample from the distribution of Press-Release timings for key c
LET DIGRAPH-TIME(c, c') ← A function that returns a sample from the distribution of Release-Press timings between keys c and c'
INITIALIZE samples ← an empty array of length SAMPLE-SIZE
INITIALIZE P ← the target password
INITIALIZE i ← 0
while i < SAMPLE-SIZE do
  INITIALIZE timings ← an empty array of the same length as P
  INITIALIZE j ← 0
  while j < 2 × length of P do
    timings[j] ← HOLD-TIME(P[j])
    timings[j + 1] ← DIGRAPH-TIME(P[j], P[j+1])
    j = j + 2
  end while
  samples[i] ← timings
  i = i + 1
end while
CALL Adversarial k-means++ using samples
```

IV. Experiments

In this section, we present the experiments we performed to evaluate our adversarial attacks. We experimented with two modalities of behavioral biometrics (keystroke dynamics and touchscreen swipes), various datasets, and multiple state of the art classification algorithms. Our experiments demonstrated that our attack algorithm was effective across all these different settings.

A. Experimental Setup

1) Protocols: Here we present the protocol we followed for collecting new data, and selecting samples for training and testing. This should shed light on the important decisions we made, but is not meant to be a comprehensive list. Instead it can be used along with the code for the data collection stage, and experiments [2] for replicating our work.

Collecting MTurk dataset: The dataset was collected on the Internet using JavaScript features. During the study we disabled typing features such as copy, paste and backspace. Users were presented the passwords in a random order and could go to the next word only after typing a given word without any errors.

We dropped any malformed samples in a pre-processing step. These could happen due to a combination of reasons that include: different behavior of browsers, differences in internet speed, or other noise as the subjects took the study simultaneously. For instance, one common scenario was when we did not receive the key-up events for every pressed character. Rather than going over every minor decision in the way we had set up the website, and the subsequent data pre-processing

[2] https://github.com/parimarjan/adversarial_keystrokes

MTurk dataset passwords	Words used by the indiscriminate adversary
mustang	mutter mumble bus fuss tryst list data iota than crane bang rang
password	pat part taste fast boss cross swat answer woman wolf bored more shard gird
letmein	lest lead beta met paytm tmux me same veil height win sin
abc123	abs fab bobcat bc 412 128 235 423 mac1 tic1
123456789	124 412 236 623 348 834 4510 1045 5612 1256 6714 1467 7816 1678 890 089

TABLE I: On the left side are the passwords from the MTurk dataset. On the right side are words that contain the digraphs from the given password. These were used to collect keystroke samples for the indiscriminate k-means++ adversary. For instance, "mutter", and "mumble" provide keystroke samples for "mu", as in the password "mustang".

step, we provide our code for these steps, so interested readers can directly refer to them for details.

Next, we describe the protocol we used for selecting samples for training and testing, and creating adversarial samples across all datasets. The training stage was used to fit the classifier models with each of the user's samples, and the testing stage was used to compute EER scores, and set the individual thresholds for each user. Then in the final adversarial stage we tested the robustness of the classifier to artificially generated samples.

There are two broad categories of classifiers used in the context of behavioral biometrics authentication: one class classifiers and two class classifiers. One class classification algorithms only use samples from the genuine user to train the model, while two class classification algorithms are also given access to some of the impostor samples. Traditionally, keystroke dynamics based authentication systems have focused more on one class classifiers because it is very impractical to expect negative samples for an arbitrary password. For instance, in the 2016 KBOC competition, only one class classifiers were used [26]. Another reason is that generally both the two class classifiers, and one class classifiers appear to give similar EER scores, so there has been no good reason to prefer two class classifiers. In our analysis in the rest of the paper, we assume an idealized two class classifier scenario in which the classifier has access to readily available impostor samples. Using our datasets, this is easy to simulate as all the users were typing the same passwords. At the same time, two class classifiers seem to work well with the touchscreen swipes features. This is because most of them are global values (like mean speed, mean gravity and mean pressure) collected for the swipe as a whole. In comparison, keystrokes features were broken down into chunks based on which letters were being typed. One of the consequences is that it is possible to get population data on these global features for an arbitrary swipe.

Genuine User Samples: In all the datasets, we follow the DSN approach of using the first half of the samples for training, and the second half for testing [19]. This makes sense because it models the realistic situation where an online classifier will use the first samples from a user to classify future samples. We also experimented with randomly dividing the samples into two equal halves - it usually produces slightly better EERs, but does not change the adversarial results that we present here.

Impostor Training Samples: This was only required for the two class classifiers. We randomly chose the same number of impostor training samples as the genuine user's training samples for each of the classifiers.

Impostor Testing Samples: For Killourhy-Maxion's DSN dataset, we followed their strategy: first four samples of every user besides the genuine user. To keep the number of positive and negative samples balanced, for the MTurk and touchscreen swipes dataset, we randomly sampled the same number of impostor samples as the genuine user's test samples.

Adversary: The Targeted K-means++ adversary used all the samples from the data set excluding the ones from the target user and the ones used for training and testing the user's classifier. For the Indiscriminate K-means++ adversary, we conducted a new MTurk study, as described before, a few months after the original study. We used all the samples from this new study. In Algorithm 2, we set the parameter "SAMPLE-SIZE" to 20000.

2) Detection Algorithms: We used the following behavioral biometrics algorithms. In particular, these include most of the classifiers used in KBOC [26], which represents the state of the art in keystroke dynamics. We also chose the classifiers to represent diverse methods - from statistical classifiers, to deep learning based networks.

One Class Classifiers:

- **Manhattan distance:** A test sample is accepted if its average feature-wise Manhattan (city-block) distance to the mean of the training set is below a threshold. If x is a test sample (of m dimensions) and μ is the mean of the training set, the distance is defined as $\sum_{i=1}^{m} |x_i - \mu_i|/m$. Killourhy-Maxion [19] had actually shown a variant of Manhattan distance, the scaled Manhattan distance, performs slightly better. This was because the scaled Manhattan distance deals better with outliers. But this is precisely the advantage of using the feature normalization techniques described below, so we found that the scaled version added nothing to the Manhattan distance classifier.
- **Gaussian:** The training samples are modeled as a Gaussian distribution based on their mean and standard deviation. If the probability of a particular test sample being in the distribution is above a particular threshold, then it will be accepted.
- **Gaussian mixture:** Here, the training samples are fitted to a Gaussian Mixture model with two com-

ponents using the EM algorithm. Then newer samples are scored based on their probability of belonging to the distribution of the training samples. We used the implementation in the python library sklearn [32].

- **One Class SVM:** We used the Support Vector Machine (SVM) implementation in sklearn [32], with radial basis function (RBF) kernel, and kernel parameter 0.9, as used in [24], [25].
- **Autoencoder and Contractive Autoencoder:** With the advent of deep learning, researchers have started using variants of neural networks in the domain of cybersecurity. One of the key structures used in the past are autoencoders and contractive autoencoders. [24], [25].

Two Class Classifiers:

- **Random Forests:** We used a model similar to the one described by Antal et al [4]. Random Forests with 100 trees was their best-performing classifier on the touchscreen swipes dataset. We used the Random Forest implementation in sklearn [32].
- **Nearest Neighbor:** Here we classify a test sample based on the majority label among a fixed number of its nearest neighbors in the training set. The neighbours are determined using Euclidean distance. We used the implementation in [32].
- **Fully Connected Neural Net:** We experimented with multiple variants of multi layer perceptron by using different hyper parameters. The network that performed the best had two hidden layers with 15 neurons each computing scores for genuine and impostor classes. There was no significant improvement in the performance of the network by increasing the number of layers or neurons per layer in the architecture of the neural network.

Monaco's Normalization Technique: Many new techniques to improve the performance of the classifiers were developed in the KBOC competition [26]. Most interesting of these were the normalization techniques developed by Monaco. The key insight of this technique was that a user's classifier could normalize future input samples based only on the genuine user's data given to it at the start. Essentially, this acts like a filtering step - and features that are too far from the mean of the genuine user's fitting data get filtered out [24], [25].

This was one of the novel techniques that helped Monaco's classifiers win the KBOC challenge. No previous results with this technique were reported for the DSN dataset - but we found that it improved performance of all the algorithms significantly. In fact, the results we report in Table II are the best reported EER scores for these classifiers. It also made the algorithms perform better against our adversarial attempts - for instance, in cases where we used a much more conservative threshold, shown in Figure 3, the classifier's performance is considerably improved with this normalization technique. Therefore, we do not even mention our results without this normalization. In a similar way, the scores output by each classifier were normalized for each user, as used by Monaco [24].

Name of Classifier	DSN EER	MTurk EER
Manhattan	0.091	0.097
SVM	0.087	0.097
Gaussian	0.121	0.109
Gaussian Mixture	0.137	0.135
Autoencoder	0.099	0.099
Contractual Autoencoder	0.086	0.099
Random Forest	0.08	0.067
k-NN	0.09	0.090
FC Neural Net	0.08	0.091

TABLE II: EER scores on the DSN and MTurk datasets

B. Results

We surveyed the algorithms proposed in the literature and implemented several of the best ones, with the aim of replicating the best existing results. Then, we devised and experimented with the adversarial agents described before to study the robustness of the proposed models against such attacks. We demonstrate that our adversarial attacks can effectively defeat all the proposed models.

1) Equal Error Rate: The EER results from our classifiers on the DSN dataset and MTurk datasets have been summarized in Table II. For the DSN dataset, these are some of the best reported scores. For instance, without the normalization technique described above, Manhattan and SVM EER scores were both around 0.15 - so it nearly doubled their accuracy. Similarly, we saw improvements in the other classifiers as well. The classifiers performed similarly also on the much bigger MTurk dataset suggesting that these are state of the art classifiers for this problem.

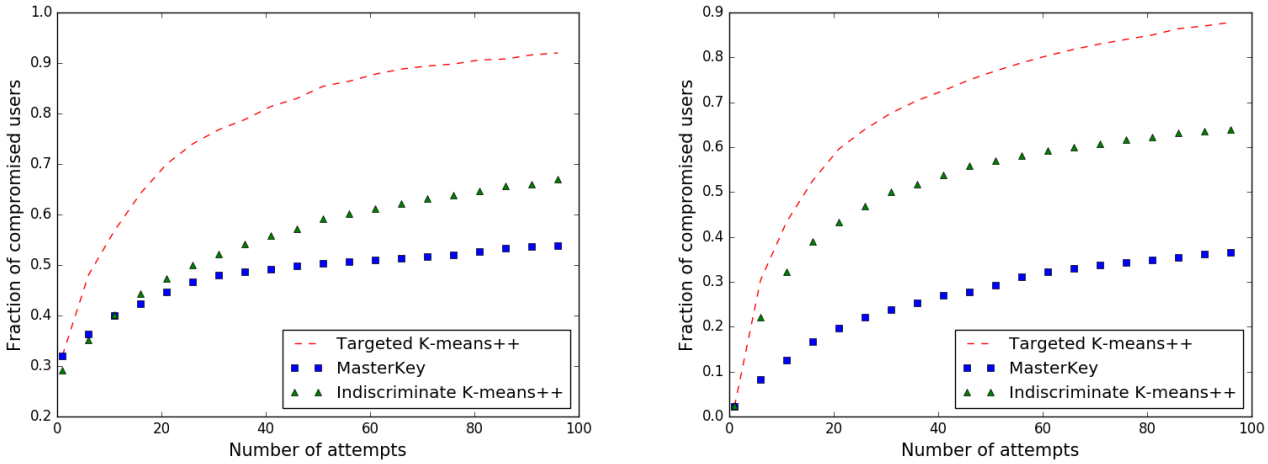
2) Keystroke Results: In this section we discuss the results of testing our adversaries on the DSN and MTurk datasets, which are summarized in Tables III, IV. We conducted the tests independently on each of the five passwords in the MTurk dataset, but for a more compact presentation, we average the results of all passwords. A few interesting highlights based on these results are given below.

MasterKey vs K-means++: Recall that the first try for all the adversaries was the mean of the available samples - so MasterKey and targeted k-means++ start at the same level of success as they utilize the same adversarial samples. But as it can be seen from the highlighted segments of the Tables III, IV and the Figure 2, the performance of our adversaries is lower bounded by the performance of the MasterKey algorithm. The tables and figures also show that Indiscriminate K-means++ also performs consistently better than MasterKey.

A significant difference between the adversaries is that MasterKey's performance improves much more slowly as compared to our adversaries. This is because it does not explore the sample space of possible typing patterns well. Its tries are derived from exploring outwards from the mean of adversarial samples by perturbing values of this initial estimate - so it does not improve much even after hundreds of tries. Meanwhile, both the k-means++ algorithms continue to compromise more users at regular intervals as can be seen in Figure 2. In

Classifier	MasterKey	Targeted K-means++
Manhattan	1: 0.55	1: 0.57
	10: 0.61	**10: 0.67**
	50: 0.61	**50: 0.86**
SVM	1: 0.65	1: 0.65
	10: 0.75	**10: 0.8**
	50: 0.78	**50: 0.92**
Gaussian	1: 0.53	1: 0.53
	10: 0.57	**10: 0.69**
	50: 0.57	**50: 0.96**
Gaussian Mixture	1: 0.67	1: 0.65
	10: 0.71	10: 0.76
	50: 0.82	**50: 0.96**
Autoencoder	1: 0.67	1: 0.63
	10: 0.73	10: 0.8
	50: 0.8	**50: 0.94**
Contractive Autoencoder	1: 0.65	1: 0.65
	10: 0.69	**10: 0.84**
	50: 0.78	**50: 0.94**
RandomForests	1: 0.06	1: 0.06
	10: 0.18	10: 0.31
	50: 0.33	**50: 0.69**
FC Neural Net	1: 0.0	1: 0.0
	10: 0.1	**10: 0.33**
	50: 0.14	**50: 0.73**
k-NN	1: 0.06	1: 0.04
	10: 0.31	10: 0.37
	50: **0.57**	**50: 0.82**

TABLE III: Fraction of users in the DSN dataset whose classifiers were compromised after 1, 10 and 50 tries of MasterKey and Targeted K-means++ algorithms for each of the classifiers we used. The bold values highlight that K-means++ outperforms MasterKey.



(a) Performance of adversaries against SVM classifier on the MTurk dataset (b) Performance of adversaries against Random Forests classifier on the MTurk dataset

Fig. 2: Comparison of Targeted K-means++, MasterKey, and Indiscriminate K-means++ adversaries over first 100 attempts. (a) shows one of the best one class classifiers, and (b) shows one of the best two class classifiers.

particular, Targeted K-means++ seems to essentially be able to compromise the security of all the users in the limit.

MasterKey also performs worse with a larger number of users as in the MTurk dataset (Table IV, Figure 2). This also seems to be a direct consequence of not exploring the sample space of key-press timings beyond the mean very efficiently. Especially since the MTurk data was collected over the Internet rather than on a single machine, it was less uniform. Therefore,

Classifier	MasterKey	Targeted K-means++	Indiscriminate K-means++
Manhattan	1: 0.272	1: 0.272	1: 0.258
	10: 0.352	**10: 0.516**	10: 0.374
	50: 0.444	**50: 0.836**	**50: 0.568**
SVM	1: 0.32	1: 0.32	1: 0.288
	10: 0.394	**10: 0.552**	10: 0.402
	50: 0.502	**50: 0.854**	**50: 0.588**
Gaussian	1: 0.306	1: 0.306	1: 0.302
	10: 0.318	**10: 0.554**	**10: 0.424**
	50: 0.344	**50: 0.88**	**50: 0.632**
Gaussian Mixture	1: 0.322	1: 0.322	1: 0.314
	10: 0.45	**10: 0.634**	10: 0.528
	50: 0.61	**50: 0.904**	**50: 0.742**
Autoencoder	1: 0.322	1: 0.322	1: 0.286
	10: 0.444	**10: 0.604**	10: 0.424
	50: 0.596	**50: 0.878**	**50: 0.642**
Contractive Autoencoder	1: 0.304	1: 0.302	1: 0.292
	10: 0.37	**10: 0.52**	10: 0.39
	50: 0.472	**50: 0.82**	**50: 0.566**
Random Forests	1: 0.022	1: 0.022	1: 0.024
	10: 0.118	**10: 0.414**	**10: 0.306**
	50: 0.274	**50: 0.782**	**50: 0.556**
FC Neural Net	1: 0.01	1: 0.01	1: 0.01
	10: 0.202	**10: 0.492**	**10: 0.544**
	50: 0.454	**50: 0.868**	**50: 0.818**
k-NN	1: 0.09	1: 0.092	1: 0.102
	10: 0.34	10: 0.55	10: 0.536
	50: 0.58	**50: 0.936**	**50: 0.806**

TABLE IV: Fraction of users in the MTurk dataset whose classifiers were compromised after 1, 10 and 50 tries of MasterKey, Targeted K-means++, and Indiscriminate K-means++ for each of the classifiers. The bold values highlight that K-means++ adversaries outperformn Masterkey.

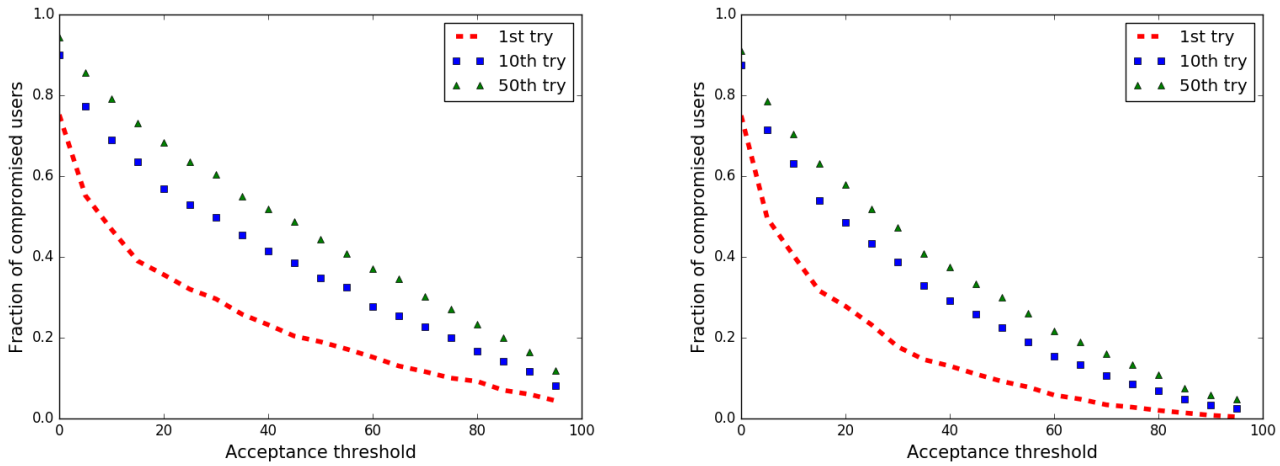
exploring around the mean of the sample space is not rewarded as much as it is in the smaller DSN dataset.

One class vs two class classifiers: For the one class classifiers, it is particularly surprising to see a majority of the user's classifiers compromised after just a single try (Tables III, IV). In comparison, the first tries are not effective at all against the two class classifiers. This was clearly because the two class classifiers had access to some samples from the impostors. It is worth noting that the classifiers only saw a small proportion of the total samples from the impostors - for instance, in the MTurk dataset, each classifier only had access to 50 impostor training samples, out of a total of over 50000 samples. This highlights the point that global EER scores are not the ideal measure for the security guarantees provided by such classifiers. Even then, the performance of the adversaries against the two class classifiers in the limit appears to be converging to the performance of the one class classifiers (Figure 2). This suggests that even these idealized two class classifiers are far from a great solution for keystrokes authentication.

Targeted K-means++ vs Indiscriminate K-means++: Despite having access to almost 20 times less data, the Indiscriminate K-means++ adversary still performed reasonably well. Even though both the adversaries start close to each other, unsurprisingly, the indiscriminate adversary was worse than the more powerful, Targeted K-means++ in the long run (Figure 2). But notice that the indiscriminate adversary never plateaus, and continues to improve steadily up to a 100 tries, reaching $55 - 75\%$ range of compromised users. This steady improvement highlights the fact that it has potential to improve further if the indiscriminate adversary had more power - for instance access to more data, or data with a longer subsequence of the password.

DSN vs MTurk datasets: In general, the results of the EER scores, and adversarial attempts, appears to be consistent across both the datasets. Therefore, we present the figures from the MTurk dataset as it is bigger and more representative.

Conservative Thresholds: In the Figure 3, we vary the acceptance threshold score for authentication to model stricter security settings. We follow the same protocol for selecting samples, training, and testing - but instead of setting the threshold at the EER point, we set it based on the scores the classifier assigned the user's test samples. We vary the threshold from the 0th to 100th percentile of these test scores



(a) Targeted K-means++ against the Manhattan classifier on the MTurk dataset(b) Indiscriminate K-means++ against the Random Forests classifier on the MTurk dataset

Fig. 3: Performance of the K-means++ adversaries against increasingly more conservative thresholds. The x-axis represents the acceptance threshold of the given classifier as a percentile of the user's test scores. For instance, $x = 50$, represents a threshold at which half the genuine user's samples were rejected

- so for instance, at the 0th percentile, all the genuine user's test samples will be accepted, and at the 100th percentile, none of these samples would be accepted. A threshold around the median level is certainly not a very usable scenario in general. But in certain high risk situations, for instance if the connection to the account was from an unknown IP address, such measures may make sense. As in the previous figure, we present the plots on the MTurk dataset, but these trends are even more pronounced on the smaller DSN dataset. In plot(a), we present the plot of the Targeted K-means++ adversary against Manhattan, one of the best one class classifiers. This shows that even with much more conservative thresholds the classifiers are still not too effective - with the acceptance threshold set at the median level, the adversary is still able to bypass nearly half the users in 10 tries. In plot(b), we consider Indiscriminate K-means++ against Random Forests, the best of the two class classifiers. This shows a similar trend to the previous figure - the indiscriminate adversary performs a little worse than the targeted adversary, but still breaks a non trivial number of users even against extremely conservative thresholds. Finally, we would like to point out that the trend shown in these two plots is also seen with different combinations of classifiers and adversaries. This shows that these classifiers should not be able to make the authentication systems arbitrarily secure on demand by changing the acceptance thresholds.

3) Touchscreen swipes dataset results: We also ran our whole suite of classifiers on the touchscreen swipes dataset, however, most classifiers performed poorly on it. This is to be expected as most of these classifiers were not used in the original paper by Antal et al. [4]. So we analyzed the performance of our adversary against the best performing one class, and two class classifiers from their original study.

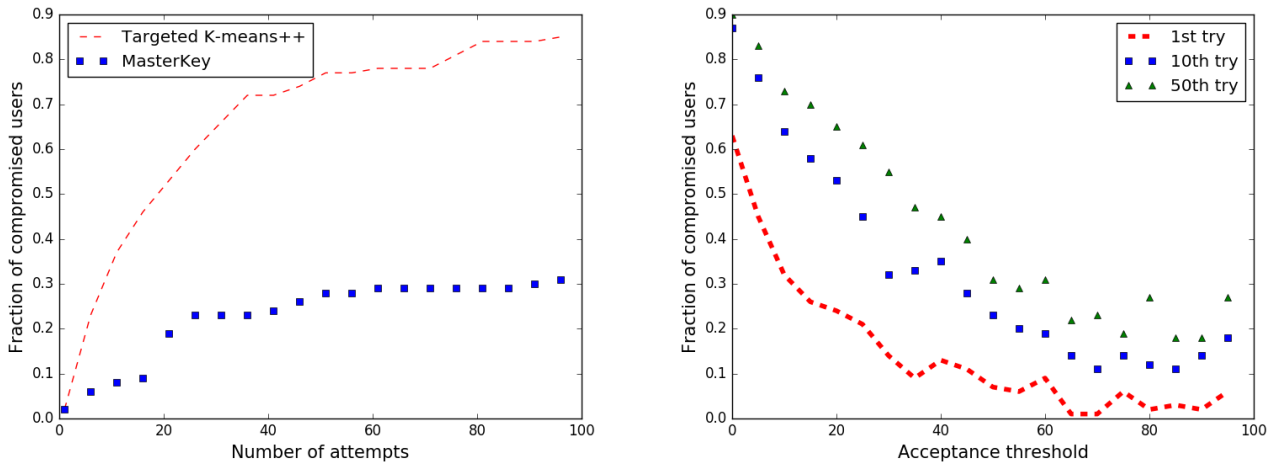
There were a couple of differences as compared to the original study [4]. We had access to a bigger size of participants from their online source [3], and we also used a different protocol for selecting training and test samples for the reasons explained in section $IV A$. But the EER results in Table V are in the same range as the ones reported in the original study [4].

Classifier Name	EER	MasterKey	Targeted K-means++
Random Forests	0.069	1: 0.04	1: 0.03
		10: 0.11	**10: 0.42**
		50: 0.32	**50: 0.79**
Gaussian Mixture	0.12	1: 0.03	1: 0.03
		10: 0.03	**10: 0.33**
		50: 0.05	**50: 0.72**

TABLE V: Summarizing the experimental results on touchscreen swipes dataset. For the adversaries, Masterkey and Targeted K-means++, the values represent the fraction of users compromised after given number of tries.

We tested the Targeted K-means++ adversary and MasterKey on this authentication system. The adversaries could be applied without any change as they operate on the vectors of feature values. Since the Indiscriminate K-means++ adversary was clearly based on specific properties of keystroke dynamics, it could not be applied to this dataset.

As can be seen by Table V, and Figure 4, the results on this dataset show the same trends as seen in the keystroke dynamics datasets before. The first try which hits the mean of the impostor samples is not very successful here. This is particularly bad for an adversary like MasterKey which stays around the mean of the distribution, and is reflected in the results in Table V. But the K-means++ adversary is quickly able to explore the sample space to find more challenging queries - and in 10 tries itself, breaks into a sizeable proportion



(a) Performance of adversaries against Random Forests classifier on the touchscreen swipes dataset.

(b) Targeted K-means++ against different acceptance thresholds of the Random Forests classifier on the touchscreen swipes dataset.

Fig. 4: Touchscreen swipes dataset versions of figures 2 and 3. We essentially see similar patterns to the results seen on the keystroke dynamics datasets.

of the classifiers as in the keystrokes dataset. And in the limit, essentially all the user's classifiers are compromised.

V. Conclusion and Future Work

Behavioral biometrics is a promising field of research, but it is not a reliable solution for authentication in its current state. We proposed two adversarial agents that require a different amount of effort from the adversary. Both attack methods performed clearly better than the previously studied attack methods in the literature and show that current state of the art classifiers add little protection against such adversaries. In the case of Indiscriminate K-means++, more than its success rate, it is worrying for the keystroke dynamics systems that such an adversary could conduct its attack without any additional cost incurred to collect samples. Past research has focused much more on improving the classifiers against naive adversaries, but this work shows that a lot more research from the adversarial perspective is required before such authentication systems can be adopted in sensitive contexts.

The design of our K-means++ adversaries utilizes a common intuition about human behavior, which is that a person's behavioral data belongs to a "cluster", rather than being absolutely unique. Thus it is natural to expect such techniques to generalize to other types of behavioral data. The results on the touchscreen touchscreen swipes dataset also supports this claim.

Of course, from a practical perspective, it is much harder to simulate an attack on a touchscreen based system, as opposed to a keystroke dynamics system, because of the diversity of the touchscreen features like pressure, finger size and so on. Unlike keystrokes - we can't just write an easily automated script to carry out such an attack. This implies that a swipes based classifier is more secure for now. But given enough motivation, it is possible that methods could be devised to bypass such limitations. For instance, such attacks could be carried out by feeding false information to the android sensors, or in an extreme example, by building a robotic arm.

Previous research has relied exclusively on the average Equal Error Rate scores across all subjects to measure the robustness of classifiers. To develop more robust behavioral biometric classifiers, it would be useful to benchmark against the adversarial agents proposed in this paper instead. For instance, one class classifiers have been the dominant method researched in the keystroke dynamics literature as they perform as well as the two class classifiers in terms of EER, while the two class classifiers are not practical because one can not expect impostor samples for arbitrary passwords. Yet, against both the adversarial algorithms, the two class classifiers performed clearly better than the one class classifiers. This suggests that a future direction of research would be to bridge the gap between the idealized and practical versions of such two class classifiers as explained in section $IV.A$.

From the adversarial perspective, one possibility for future work would be to extend these methods to free text based classifiers. Free text classifiers utilize a continuous stream of input text, as opposed to fixed text passwords, in order to classify keystroke patterns. This leads to differences in the features and algorithms that are utilized for these classifiers. But conceptually, the Indiscriminate K-means++ adversary should be well suited to generate adversarial samples against free text classifiers as well.

Acknowledgment

The authors would like to thank David Mazieres for helpful discussions and support during this project.

References

[1] A. K. Abdul Serwadda, Vir Phoha, "Using global knowledge of users' typing traits to attack keystroke biometrics templates," in *Thirteenth*

ACM Multimedia Workshop on Multimedia and Security, New York, NY, USA, 2011, pp. 51–60.

[2] A. A. E. Ahmed and I. Traore, "A new biometric technology based on mouse dynamics," *IEEE Transactions on dependable and secure computing*, vol. 4, no. 3, p. 165, 2007.

[3] M. Antal. (2016, October) Eysenck Personality Questionnaire Android platform. http://www.ms.sapientia.ro/~manyi/personality.html. [Online; accessed 30-November-2017].

[4] M. Antal and L. Z. Szabó, "Biometric authentication based on touchscreen swipe patterns," *Procedia Technology*, vol. 22, pp. 862–869, 2016.

[5] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Philadelphia, PA, USA, 2007, pp. 1027–1035.

[6] M. Barreno, B. Nelson, A. Joseph, and J. D. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, no. 2, pp. 121–148, November 2010.

[7] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 387–402.

[8] J. Brodkin. (2012) 10 (or so) of the worst passwords exposed by the LinkedIn hack. https://arstechnica.com/information-technology/2012/06/10-or-so-of-the-worst-passwords-exposed-by-the-linkedin-hack/. [Online; accessed 30-November-2017].

[9] C. Burt. (2016) TypingDNA enables easy identity verification for web apps with typing biometrics. http://www.biometricupdate.com/201612/typingdna-enables-easy-identity-verification-for-web-apps-with-typing-biometrics. [Online; accessed 30-November-2017].

[10] J. Chang. (2014) Kickstarter hack attack leaks user passwords. http://abcnews.go.com/Technology/passwords-email-addresses-leaked-kickstarter-hack/story?id=22553952. [Online; accessed 30-November-2017].

[11] Y. Deng and Y. Zhong, "Keystroke dynamics user authentication based on gaussian mixture model and deep belief nets," in *Int. Sch. Res. Not.*, ser. p. e565183, October 2013.

[12] M. O. Derawi, C. Nickel, P. Bours, and C. Busch, "Unobtrusive user-authentication on mobile phones using biometric gait recognition," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*. IEEE, 2010, pp. 306–311.

[13] Y. Ding and P. Horster, "Undetectable on-line password guessing attacks," *SIGOPS Oper. Syst. Rev.*, vol. 29, no. 4, pp. 77–86, Oct. 1995. [Online]. Available: http://doi.acm.org/10.1145/219282.219298

[14] M. Frank, R. Biedert, E.-D. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *Information Forensics and Security, IEEE Transactions on*, vol. 8, no. 1, pp. 136–148, 2013.

[15] S. Furnell, "Continuous user identity verification using keystroke analysis," in *Proceedings of International Conference on Multimedia Communications, Southampton*, 1995, pp. 189–193.

[16] D. Goodin. (2013) Why LivingSocials 50-million password breach is graver than you may think. https://arstechnica.com/security/2013/04/why-livingsocials-50-million-password-breach-is-graver-than-you-may-think/. [Online; accessed 30-November-2017].

[17] D. Guccione. (2016) How Multifactor Authentication Can Play a Role in the Cybersecurity National Action Plan. http://www.nextgov.com/technology-news/tech-insider/2016/05/how-multifactor-authentication-can-play-role-cybersecurity-national-action-plan/127945/. [Online; accessed 30-November-2017].

[18] S. Hashiaa, C. Pollettb, M. Stampc, and M. Hall, "On using mouse movements as a biometric," 2005.

[19] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *IEEE/IFIP International Conference on Dependable Systems Networks*, 2009, pp. 125–134.

[20] P. Lilly. (2016) Google Project Abacus To Replace Android Passwords With Biometric And Environmental Trust Score. https://hothardware.com/news/google-project-abacus-replace-passwords-trust-score. [Online; accessed 30-November-2017].

[21] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 641–647.

[22] W. Melicher, B. Ur, S. M. Segreti, S. Komanduri, L. Bauer, N. Christin, and L. F. Cranor, "Fast, lean and accurate: Modeling password guessability using neural networks," in *Proceedings of USENIX Security*, 2016.

[23] R. Miller. (2016) UnifyID wants to bury the password once and for all. https://techcrunch.com/2016/09/12/unifyid-wants-to-bury-the-password-once-and-for-all. [Online; accessed 30-November-2017].

[24] J. V. Monaco. (2016) Code for submissions to the Keystroke Biometrics Ongoing Competition. https://github.com/vmonaco/kboc. [Online; accessed 30-November-2017].

[25] ——, "Robust keystroke biometric anomaly detection," *arXiv preprint arXiv:1606.09075*, 2016.

[26] A. Morales, J. Fierrez, R. Tolosana, J. Ortega-Garcia, J. Galbally, M. Gomez-Barrero, A. Anjos, and S. Marcel, "Keystroke biometrics ongoing competition," *IEEE Access*, vol. 4, pp. 7736–7746, 2016.

[27] M. Muaaz and R. Mayrhofer, "Smartphone-based gait recognition: From authentication to imitation," *IEEE Transactions on Mobile Computing*, 2017.

[28] B. Nelson, B. I. Rubinstein, L. Huang, A. D. Joseph, S. J. Lee, S. Rao, and J. Tygar, "Query strategies for evading convex-inducing classifiers," *Journal of Machine Learning Research*, vol. 13, no. May, pp. 1293–1332, 2012.

[29] B. Nelson, B. I. Rubinstein, L. Huang, A. D. Joseph, and J. Tygar, "Classifier evasion: Models and open problems," in *International Workshop on Privacy and Security Issues in Data Mining and Machine Learning*. Springer, 2010, pp. 92–98.

[30] P. Olson. (2016, March) Forget passwords. now banks can track your typing behavior on phones. [Online]. Available: http://www.forbes.com/sites/parmyolson/2014/08/18/forget-passwords-now-banks-can-track-your-typing-behavior-on-phones/

[31] A. S. Osborn, *Questioned Documents*, 1910.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[33] K. A. Rahman, K. S. Balagani, and V. V. Phoha, "Making impostor pass rates meaningless: A case of snoop-forge-replay attack on continuous cyber-behavioral verification with keystrokes," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*. IEEE, 2011, pp. 31–38.

[34] A. Serwadda and V. V. Phoha, "Examining a large keystroke biometrics dataset for statistical-attack openings," *ACM Transactions on Information and System Security (TISSEC)*, vol. 16, no. 2, p. 8, 2013.

[35] D. Stefan and D. Yao, "Keystroke-dynamics authentication against synthetic forgeries," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*. IEEE, 2010, pp. 1–8.

[36] N. Y. Ted Dunstone, *Biometric System and Data Analysis: Design, Evaluation, and Data Mining*. Springer, 2008.

[37] C. M. Tey, P. Gupta, and D. Gao, "I can be you: Questioning the use of keystroke dynamics as biometrics," 2013.

[38] N. Zheng, A. Paloski, and H. Wang, "An efficient user verification system via mouse movements," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 139–150.