Guessing Attacks on User-Generated Gesture Passwords

CAN LIU, GRADEIGH D. CLARK, and JANNE LINDQVIST, Rutgers University

Touchscreens, the dominant input type for mobile phones, require unique authentication solutions. Gesture passwords have been proposed as an alternative ubiquitous authentication technique. Prior security analysis has relied on inconsistent measurements such as mutual information or shoulder surfing attacks. We present the first approach for measuring the security of gestures with guessing attacks that model real-world attacker behavior. Our major contributions are: 1) a comprehensive analysis of the weak subspace for gesture passwords, 2) a method for enumerating the size of the full theoretical gesture password space, 3) a design of a novel guessing attack against user-chosen gestures using a dictionary, and 4) a brute-force attack used for benchmarking the performance of the guessing attack. Our dictionary attack, tested on newly collected user data, achieves a cracking rate of 47.71% after two weeks of computation using 10^9 guesses. This is a difference of 35.78 percentage points compared to the 11.93% cracking rate of the brute-force attack. In conclusion, users are not taking full advantage of the large theoretical password space and instead choose their gesture passwords from weak subspaces. We urge for further work on addressing this challenge.

CCS Concepts: • **Security and privacy → Authentication**; • **Human-centered computing → Gestural input**;

Additional Key Words and Phrases: free-form gesture, guessing attack, password space, dynamic time warping

ACM Reference format:
Can Liu, Gradeigh D. Clark, and Janne Lindqvist. 2017. Guessing Attacks on User-Generated Gesture Passwords. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 1, Article 3 (March 2017), 24 pages.
DOI: https://doi.org/10.1145/3053331

1 INTRODUCTION

Touchscreens are the dominant way to interact with new and emerging computing technologies, in particular mobile devices. This interaction method requires user-centric authentication techniques to deal with the usability constraints of mobile use. Unlocking methods need to answer competing requirements of mobile interaction [15, 20, 26]: lack of visual attention to the screen, frequent unlock attempts, and relatively short length of a session.

Gesture passwords are a recently proposed authentication method for mobile devices that is attracting interest due to their ability to satisfy these usability criteria [1, 2, 13, 30, 34, 37, 48–50]. A gesture in the context of this work, as well as previous work [34, 50], is defined as a series of two-dimensional lines drawn on the surface of a touchscreen with one or more fingers. This gesture is then encoded as a set of X and Y coordinates and compared to a template for authentication.

This work is supported by the National Science Foundation under Grant Number 1228777. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Gradeigh D. Clark was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program. We thank Joseph Bonneau and Shridatt Sugrim for helpful discussions.
Author's address: C. Liu, G. Clark and J. Lindqvist, Electrical & Computer Engineering Building, 94 Brett Road, Piscataway, New Jersey, 08854; email: can.liu@rutgers.edu, gradeigh.clark@rutgers.edu, janne.lindqvist@rutgers.edu.
 2474-9567/2017/3-ART3 $15.00
DOI: https://doi.org/10.1145/3053331

This emerging method has been studied and found to have some advantages. Prior work has shown that gestures can be performed quickly – 22% faster than text passwords [50], their memorability is unique under multi-account interference [50], they have a high amount of information content and complexity [34], and biometric features are distinct enough between users of the same gesture to be used to uniquely identify users [30, 33, 37]. User choices for mobile authentication are often driven by personal preference and usability criteria rather than overt security concerns. For example: users prefer the Android Pattern Unlock over PINs because of personal comfort, despite being able to enter PINs faster [42]. This was concluded independently by users without consideration to security, of which Android Pattern Unlock has been shown to be weaker than a 3-digit PIN [38]. Users have expressed preferences towards using gesture passwords when presented with the option over other methods [30, 50].

Gesture passwords share similarities to graphical passwords, which have been studied in detail over the past decade and have gained prominence with the Android Pattern Unlock. However, there are key differences between gesture passwords and graphical passwords. Graphical passwords present a password as a sequence of predefined points (e.g. Android Pattern Unlock [38], Draw-A-Secret (DAS) [16]), as an image (e.g. Story [11]) or an area on the screen that permits user access through matching points (e.g. PassPoints [46]). Authentication with graphical passwords resembles text-based passwords in that the challenge is a *matching problem* that allows for the password to be exactly reproduced. Unlike text and graphical passwords, gesture passwords are defined as a *recognition problem* [10, 34]. A gesture password cannot be exactly matched: it must be identified despite not being input by the user the same way every time.

Although gesture passwords satisfy usability criteria and can be preferential to other unlocking methods, there are open questions about how secure gesture passwords are. Most security research on gesture passwords focuses on procedural changes to authentication systems, a simple example being increasing the number of features used in authentication. Direct studies of gesture security have used information theory [34, 50], shoulder surfing [2, 13, 33, 34], or equal error rates [1, 2, 13, 30, 34, 48–50] between legitimate users and attackers trying to emulate the gesture. The threat model in these scenarios is an attacker performing an observation of the gesture and trying to repeat it.

Another method, unexplored so far with gestures, is automated password guessing. This method is known as password cracking. The best method for modeling cracking behavior is guessability, a metric that quantifies a given password's security by counting how many guesses it takes an attacker to crack it [17, 38, 43, 44]. This metric is useful chiefly for the reason that it maps to attacker behavior better than idealized, theoretical techniques [39]. Guessability has been popularly employed to measure the security of text passwords, but not gesture passwords. For text passwords, many studies have shown that user password choice is not random but rather falls into a concentrated subset [5, 6, 14, 23]. Attackers use leaked dictionaries to perform *guessing attacks*, wherein attackers rank passwords from most likely to least likely and guess passwords in this order. This concentrated subset is referred to as the *weak subspace* since it is the weakest portion of the password set at resisting cracking attempts.

No published work on gesture security has discussed how resistant these passwords are to automated guessing attacks. We are considering a threat model where an attacker can perform multiple guesses quickly at a terminal without being locked out by the device. This is not uncommon, for example, persistent security flaws in the hardware of mobile devices can still allow for the auto-lock to be bypassed, as what happened for the iPhone [7]. There is also the attack scenario where the encrypted password data is leaked from the device, also bypassing the auto-lock. The main reason these types of attacks have not been done is because of the difficulty involved in automating an attack strategy against the recognition problem for gesture passwords. Another problem for gesture guessability is that there are no leaked gesture passwords to form a dictionary that can be ranked by probability of appearance.

Guessability has also not been applied to gesture passwords because the recognition problem is more computationally expensive and time consuming compared to the matching problem. Matching problems are quickly checked – the attacker tries text password 'abcd' and moves on to '1234' when 'abcd' does not work. Recognition

problems require an attacker to implement an additional step in their guessing attack – the attacker assumes the gesture password is a circle, and generates N slightly different circle trials before moving on to assuming the gesture password is a triangle and creating N attempts at that. The extra step is generating the N attempts per guess, which vastly slows down cracking.

Another issue affecting guessing attacks is that the recognition problem is dependent on the recognizer, of which considerable work has been done in implementing such *gesture recognizers* [10] for various environments. Gesture recognizers perform mathematical transformations on raw gesture coordinate data in order to standardize the input and to recognize it – these transformations can affect how guessing attacks are done. We chose Dynamic Time Warping (DTW) as our recognizer in this paper. We selected this recognizer because most gesture password studies and implementations [1, 2, 13, 30, 48, 49] use it. This does not limit our analysis – we derive a methodology based on using DTW as the similarity measure. This can be extended to virtually all recognition algorithms since most require a similarity step at some point (e.g. LCS, Cosine, Edit or Euclidean Distance).

Enumerating the size of the full space, the weak subspace, and being able to assign guessability values for a gesture password is valuable because it allows – uniquely – for direct numeric comparisons of security between gesture passwords and text passwords. Other metrics, such as Equal Error Rates, are often dataset dependent and are not applicable to the matching problem. This numeric metric allows for authors working on changes to gesture security systems to compare results and findings between each other as well. Usable gesture security work can use these methods to warrant claims about how modifications to human behavior, such introducing password policies, affects the security of gesture passwords.

In this paper, we present the first work on analyzing the guessability of gesture passwords. The major contributions are as follows:

Based on the largest set of gesture passwords ever assembled in research, we present a comprehensive analysis on the weak subspace for gesture passwords. An analysis of previous work [50] revealed a user bias towards choosing shapes and letters as gesture passwords. Extending this idea, we explore the weak subspace for gesture passwords revolving around collections of gestures with common meanings. By analyzing 529 gestures from previous studies [34, 50], we identified 407 gestures that were selected by users because there is some cultural or social meaning behind them. We identify and split the weak subspace into seven groups among these gestures: digits, letters, geometric shapes, special characters, mathematical symbols, music symbols, and mathematical functions. No paper we have identified in the literature on gesture passwords explores this.

We used the weak subspace to build a novel dictionary attack method against gesture passwords. Using the seven weak subspace groups we identified, we constructed likely gestures an attacker might try based on how often a given gesture appears in the datasets we collected. To the best of our knowledge, this paper presents the first dictionary attack method for recognition-based authentication.

We developed a brute force attack using symmetry. In addition to a dictionary attack, we provide a symmetry-based brute force attack with ideas rooted in a DAS attack [36] but updated and implemented for gesture passwords. We use this as a benchmark to demonstrate the effectiveness of the dictionary attack and show that our dictionary attack can crack 35.78 percentage points more gestures than the brute force method on **a new dataset from 109 participants, collected for this study.**

We developed a novel method to analyze the full theoretical password space of gesture passwords and estimate the size of the weak subspace for gesture passwords. The size of the password space refers to how many different passwords are possible. The challenge here for gesture authentication is the recognition problem. We contribute a key insight that the gesture recognizer limits the very large gesture password space down to a smaller number of discrete possibilities. We contribute a concise methodology for how to approximate the size of this space as a function of two types of preprocessing methods and reasonable parameter choices. We estimated the weak subspace sizes for gestures to be 82.1 bits and 81.7 bits, depending on the orientation preprocessing method selected for the gesture recognizer.

2 RELATED WORK

To put our work in context, we present related work for: studies on gesture authentication, studies on password guessability for text passwords and graphical passwords, and conclude with a review of the existing methods for attacks against gesture authentication systems.

Gesture Authentication: Gesture passwords that rely on user-chosen secrets form the basis of our work. Although gesture-based authentication systems can be implemented in various ways [10], the research community [1, 2, 37, 48, 49] has mostly focused on implementing recognizers with Dynamic Time Warping (DTW) [24]. We note that most of these works omit the details how gestures were preprocessed, an important point in our work because preprocessing methods change the size of the password space for gestures. Free-form gestures are also used in a system based on cosine similarity [34, 50]. Other work uses a combination of pre-defined gesture strokes on the back of the device as a password [13].

Other gesture systems focus on pre-defined gesture sequences that derive security from behavioral biometric features rather than a secret. One system examined the feasibility of pre-defined multitouch gesture and found that the uniqueness of biometric features, like hand size and finger length, can be used for authentication [30]. Similarly, implicit patterns on how people unlock their devices could be used for authentication [12]. Importantly, both of these previous approaches also use DTW as recognition method. GEAT [33] authenticates users based on finger velocity, device acceleration and stroke time.

Password Guessability: Text-based password guessability is a well-studied problem that derives its roots from biases in the user-chosen distribution of passwords. Early work examined 3289 text passwords and found that 86% of the passwords had structural overlaps: short in length, only contained lowercase letters or digits, and were common dictionary words [23]. A large-scale study on web passwords judged user passwords to be of poor quality and revealed overlap in user passwords across multiple accounts [14]. To combat the weakness of entropy, guessing entropy has been proposed to model the practical attacker behavior and quantify password security [6]. An analysis of 70 million passwords shows a clear bias in user-chosen distribution of text passwords [5]. An analysis of popular text password cracking methods reveals that quantifying password strength with a single cracking algorithm is not reliable [39]. Recently, a neural network model of human-chosen passwords was proposed, which resulted in an efficient and highly compressed password guessing method [22]. An password strength estimator - zxcvbn - can estimate the current best-known attacks with 1.5 MB compressed data and up to 10^5 guesses [45].

There is also work on the guessability of graphical passwords. With Draw-A-Secret (DAS) [16], people were found more likely to select symmetric graphics as passwords, reducing the size of the effective password space [40]. This was the first work to push forward the idea that the size of the graphical password space can be demonstrably reduced. Android's Pattern Unlock, which is a special case of DAS, was shown to skew the user-chosen distribution based on starting points and crossing patterns [38]. Instead of grid cells, Pass-Go [35] used intersections on a grid as anchors, with a study showing that 40% of users chose vertically or horizontally symmetric passwords and that 49% of the passwords are alphanumeric or well-known symbols [35].

Besides recall-based schemes, graphical passwords based on human ability to recognize images, also have weak subspaces based on user choice. Passfaces [8] presents a panel of human faces and a user creates a password by clicking a sequence of images. A field study on it showed that users tend to select the faces of their own race and the weakest 25% of user passwords were cracked in 13 attempts [11]. Instead of using human faces, Story [11] offers the user a panel of generic images to select as their password. Although the user choice issue is less severe than Passfaces, there is still a selection preference between genders and the weakest 25% passwords were guessed in 112 attempts [11]. PassPoints [46] is a cued-recall graphical password or click-based graphical password [4], with the major weaknesses being hotspots [41] and patterns [9]. Hotspots are the image areas or points that users are more likely to choose and the latest attacks based on them using Markov models can crack 36% of the

PassPoints passwords by 2^{31} guesses. [41]. Patterns are simple shapes that are likely to be chosen as PassPoints passwords: around 80% of PassPoints passwords fall into primarily defined simple shapes [9].

Based on these prior works in text passwords and graphical passwords, we can assume that the user-chosen distribution of gesture passwords could be similarly skewed. However, it has yet to be shown systematically.

Attacks Against Gesture Authentication: So far in the literature, attacks against gesture authentication have been performed by participants trying to emulate gestures based on a variable amount of knowledge about the gesture provided by researchers [1, 2, 34, 37, 48, 49]. These types of attacks are generally referred to as shoulder surfing attacks. Smudge attacks [3] were used on touchscreens to identify user passwords based on oil patterns left by their fingers. There is also a robot-based attack [32] against an touchscreen pattern authentication scheme [12], where the robot is able to emulate human patterns of use on touchscreens. However, this robot attack does not provide guessing attacks against human-chosen secrets such as gesture passwords.

In summary: there is no analysis of the effects of preprocessing methods on the size of the gesture password space. Our work explains the effects of preprocessing steps on the size of the password space. We provide a method to evaluate the security of gesture passwords through enumerating the theoretical password space. Considerable work has shown weak subspaces for text and graphical passwords, but we are the first to demonstrate the weak subspace for gesture passwords. Security of gesture passwords has been measured with mutual information and equal error rates. We provide a method more closely aligned with real world attacker behavior through guessing attacks. We present an efficient dictionary attack method as well as a brute-force method for attacking, neither of which has been demonstrated in prior work on gesture passwords.

3 METHOD

In this section, we introduce gesture datasets used in our analysis. Then, we present a generalized procedure for gesture recognition based on DTW. We propose a method for estimating the sizes of full and weak subspaces of gesture passwords. Finally, we present our algorithms for performing dictionary and brute force attacks.

3.1 Data Acquisition and Classification

Effective guessing attacks require data about user choices. The most common gesture passwords that users select represent the best options to try. Before building the machinery required to perform guessing attacks, we first need a set of data to derive insights from. A sample set is required for the attacker to use when attacking, and a test set is needed to evaluate the efficiency of our attack algorithms.

We aggregated two sets of data for these purposes. We labeled the sample set from which we derive our dictionary as the Dictionary dataset: it is composed of three gesture datasets, obtained from previous work on gesture passwords [34, 50]. It was collected from 232 users. The testing set, which we run our attack algorithms on, is called the Test dataset. It is composed of two datasets of gestures that we have newly obtained from 109 volunteer participants.

3.1.1 Dictionary Dataset. Previous work on user-generated gesture passwords classified created gestures into six rudimentary groups: Digit, Shape, Lines, Letter, Symbol, and Words [50]. We define our groups as:

- Digit: 0,1,2,3,4,5,6,7,8,9;
- Geometric Shape: triangles, circles, cylinders, etc.;
- Letter: upper and lower case of 26 letters;
- Mathematical Function: basic signal functions.
- Mathematical Symbol: characters used in mathematics;
- Music Symbol: treble clef, simple notes, etc;
- Special character: keyboard special characters.

Table 1. Summary of free-form gestures in the Test dataset (General). "Weak Subspace Feature" are the free-form gestures that fall into the weak subspace groups in Dictionary dataset. "Symmetric Feature" are the gestures that have horizontal or vertical symmetry. "Weak & Symmetric" are the gestures that have both of the two features. The "Weak Subspace Feature" size (128 gestures) is larger than "Symmetric Feature" size (83 gestures). It verifies the analysis of the Dictionary dataset, where the coverage of weak subspace features in gestures are wider than symmetric features.

Gesture Type	Unistroke	Multistroke	subtotal
Weak Subspace Feature	78	50	128
Symmetric Feature	36	47	83
Weak & Symmetric	34	29	63
Any Feature	109	109	218

In our analysis, we merged the Lines and Shape groups into the Geometric Shape group. We then extended the Symbol group by breaking it into specific meanings for four new groups: Math function, Math symbol, Music symbol, and Special character. We opted to remove the Word group because, after aggregating all the different datasets, we concluded that it does not appear as frequently as posited in prior work [50].

In the Dictionary dataset, we found 407 out of 529 gestures as ones that were created by the participants because they have some semantic meaning. These gestures, that we hypothesize are chosen because of their meaning, are what we define to be the weak subspace of gestures. In addition to these gestures, there were 122 gestures in the Dictionary dataset and 90 gestures in Test dataset (General) that were not part of any of the seven groups. Since these gestures do not have obvious meanings or is not obvious whether they would be reused by other people based on our data, these gestures did not receive any specific treatment.

3.1.2 Test Dataset. We conducted a user study to collect the Test dataset. The study was approved by the Institutional Review Board (IRB) of Rutgers University. The experiment consisted of collecting two datasets: Test dataset (General) and Test dataset (Weak).

Data Collection. The Test dataset (General) is used to test the efficiency of the guessing attack. We asked participants to create gesture passwords for a user account without any instruction as to what to choose. We asked each participant to create one unistroke gesture and one multistroke gesture. Table 1 shows that we collected 109 unistroke and 109 multistroke gestures from the participants. 136 of the user-generated passwords (78 unistroke and 58 multistroke) fall into the weak subspace groups identified in Section 3.1.1 above.

The Test dataset (Weak) is used to tune the performance of the attacking algorithm. Our guessing attack needs information about the most common variations a human might add when performing a gesture password. Our guessing attack needs to create N trials of a single gesture with only slight differences between subsequent attempts – an example would be creating a rectangle of slightly longer width each time for 100 tries. The attacking algorithm needs an idea of when to stop making adjustments to a gesture beyond a point that humans do not do – in the rectangle example, this might be making the rectangle the full width of the screen. This means we need a large number of samples to determine parameters for the variable ways people draw a given gesture in each one of the groups in Section 3.1.1. We collected Test dataset (Weak) to meet this purpose. The weak subspace contains 84 specific gestures, we asked participants to give us trials of 30 out of the 84 gestures. Totally, we collected 3270 gestures in Test dataset (Weak).

To mimic typical password creation tasks, we asked participants to first create a gesture password and then recreate it to confirm it.

Participants. We recruited participants with flyers on our university campus. We required the participants to be 18 years old or over and familiar with touchscreen devices. We recruited 109 participants with ages ranging

Table 2. Relative frequencies of seven groups of weak gestures in Dictionary dataset and Test dataset (General). It shows that the relative frequencies in two independent datasets are very similar. It means that the two datasets are similar to each other and that the size of each dataset is not significantly affecting how often each group appears.

Group Name	Dictionary Dataset	Test Dataset (General)
Digit	9.1%	9.4%
Geometric Shape	44.5%	45.3%
Letter	28.0%	28.9%
Math Function	9.1%	10.2%
Math Symbol	4.4%	5.5%
Music Symbol	2.7%	0.8%
Special Character	2.2%	0%

from 18 to 45 (Mean=20.67, SD=3.46). 50 are male and 59 are female. 88 of them were pursuing an undergraduate degree, 6 of them were pursuing graduate degrees, and the remaining 15 had a graduate degree.

Apparatus. The gesture data was collected on a Google Nexus 10 tablet with Android 4.4.2.

3.1.3 External Validity. The different sizes of the Dictionary dataset and the Test dataset (General) might affect how commonly a weak subspace group appears. This could be a problem if this were true because it would affect the generalizability of our analysis – it is possible that at very different sizes of datasets that one or more of the seven groups could be over or underrepresented in our data. At the very least, it can affect the ability to draw valid comparisons between the Dictionary and Test (General) datasets. We computed the relative frequencies of the seven groups of weak subspace gestures in both the Dictionary dataset and Test dataset (General). The relative frequencies of one group of weak subspace gestures are calculated as in Eq.(1).

$$\text{Relative frequency of group} = \frac{\#\ of\ weak\ group}{\#\ of\ entire\ weak\ subspace} \tag{1}$$

Table 2 shows that the relative frequencies of the Test dataset (General) are close to that of the Dictionary dataset despite a different sample size and different participants over separate studies. We can conclude that we are targeting similar distributions of gesture groups here and that they are suitable dictionary and attack targets, separately. We can conclude that the Test dataset (General) is a good attack candidate for the Dictionary dataset.

We are careful to note that the demographic breakdown of the Dictionary set and the Training set is preferential to people who work on campuses and college students. This affects our ability to make strong statements about how representative these group categories are when applied to the general population. It is possible that age, education, or nationality can be factors that affect the size of each gesture group. This is work that would have to be considered in the future that can be easily performed using the attack methodologies outlined by our work.

3.2 Gesture Recognition Approach

In this section, we will present the gesture recognition approach used in our analysis. First, we introduce the preprocessing procedures and its effects on the full space. Then, we will introduce Dynamic Time Warping (DTW) as the recognition method.

3.2.1 Preprocessing. People cannot exactly draw the same gestures on a touchscreen. To improve recognition accuracy, various preprocessing methods are often used to prevent relative (minor) differences between two gestures from obfuscating how similar the gestures are. The purpose of our work is to analyze the independent effect of preprocessing steps. Therefore, we want to analyze as many processing steps as possible. These independent preprocessing steps can be added or removed based on design requirements. In summary, our analysis

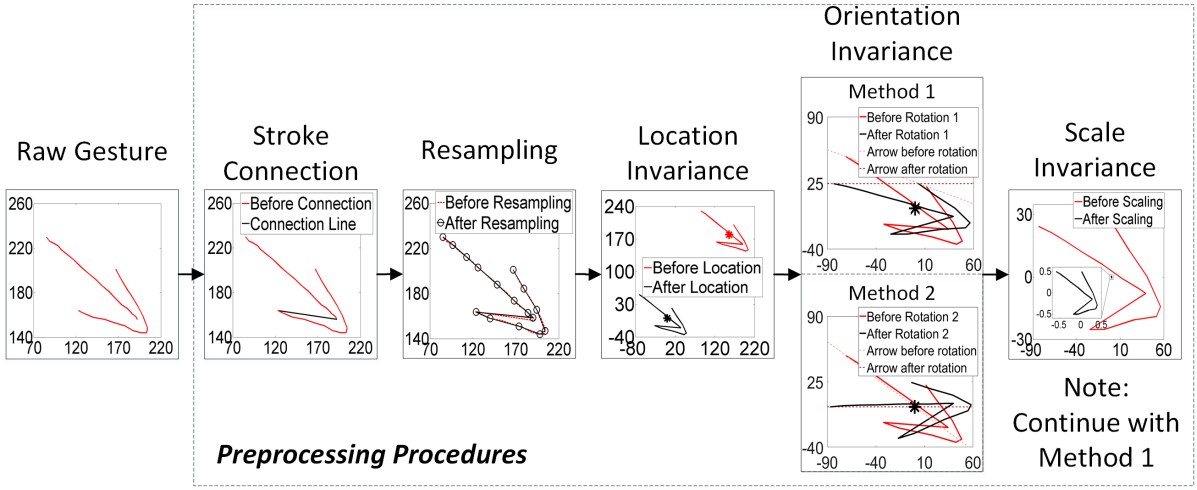

Fig. 1. The preprocessing procedures are generic steps in gesture recognition. First, user has inputted a gesture on the touchscreen (Raw Gesture). Stroke Connection: a multistroke gesture's strokes are connected to one sequence by time order. Resampling: the gesture is resampled to a fixed number of sample points. Location Invariance: the gesture is translated to the origin point. Orientation Invariance: the gesture is rotated to the same direction. Scale Invariance: the gesture is scaled to the normal size.

on preprocessing methods is applicable to approaches that measure the similarity between time series. Figure 1 shows a generic preprocessing procedure. There are five steps: 1) Stroke connection; 2) Resampling; 3) Location invariance; 4) Rotation invariance; 5) Scale invariance.

Stroke Connection. Unistroke gestures, which consist of a single continuous stroke to draw (for example, the number '8'), are easy for recognition algorithms to handle – it is a single sequence of (x,y) data coordinates. Most, if not all recognition algorithms, work best on this type of data. Multistroke gestures, which require more than one stroke to draw (for example, a # symbol), can complicate the recognition problem. These gestures are represented as multiple, separate pairs of coordinates in the Dictionary dataset. Using most recognition algorithms on these as-is would require matching every sequence exactly. This can lead to increased computation since there are $M!$ combinations to consider, where M is the number of strokes. To simplify this problem, we connect gesture strokes end-to-end as displayed in Figure 1. This transforms a multistroke gesture into a single stroke gesture and now only requires a single run of the recognition algorithm, reducing $M!$ combinations down to 1. However, if a user draws the multistroke gesture in the wrong order, this is considered to be incorrect. The effect of stroke connection is that the gestures can only have one continuous trace.

Resampling. Gestures that are sampled from the touchscreen all have coordinate sequences that are of variable length. This is a function of the sampling rate and the speed by which the user enters the gesture. Resampling gestures to a constant value removes variation and simplifies analysis. The question is, then, what is the best resampling rate to use in our analysis? The Resampling step of Figure 1 shows an illustrative example of this process. By examining the effect of different resampling rates (R=2, 4, 8, 16, 32, 64, 128) on the Equal Error Rate (EER) in the Dictionary dataset, we found that after $R = 16$, the EER values are both low and stable. We use EER to evaluate whether the parameter values have an impact on recognition accuracy. We do not use it to determine or optimize the best values based on specific datasets. The selection of suitable parameter values can significantly influence recognition performance. For example, if R is too small, information contained in a circle will be lost

and may become indistinguishable from a square. The effect of resampling is that the number of sample points is fixed to 16 and sequences with other lengths will not be considered.

Location Invariance. When a user draws a gesture on the screen, they rarely draw it at exactly the same position every time. The solution to this is to make the gesture location invariant by translating the centroid of each gesture to the origin (0,0) point of a Cartesian coordinate system. The Location invariance step in Figure 1 gives an example of this. The effect of location invariance is that the centroid is kept constant at the origin.

Orientation Invariance. Users do not draw their gestures at the same orientation every time – the centroid of the gesture may be rotated slightly off center. The solution to this problem is to make the orientation invariant. There are two methods used in literature for removing orientation variance. The upper and lower diagrams for Orientation Invariance in Figure 1 shows examples of these two methods.

Method I: Rotate the gesture until the direction vector, directed from the first point to the last point, is parallel with the x-axis.

Method II: Rotate the gesture until the direction vector, directed from the centroid to the first point, is parallel with the x-axis [47].

A difference between the two methods is that only the first and last points of gesture are involved in rotating the gesture in Method I whereas all the gesture points are involved in rotating the gesture in Method II. This could lead to a difference when it comes to resisting attacks, and therefore needs to be studied in detail.

Scale Invariance. Users rarely draw their gestures in the same size every time. As in prior steps, the solution is: Scale invariance, as shown in Figure 1, divides the gesture sequence by the range (the difference between minimum and maximum) of the gesture's X and Y coordinates for each respective coordinate pair [34, 50]. The effect of scale invariance is that the gesture is bounded in a square $(-0.5, 0.5)$.

3.2.2 Recognition Method: Dynamic Time Warping. This section briefly introduces Dynamic Time Warping (DTW) [24], which is used as our gesture recognition algorithm. We then describe the Sakoe-Chiba Band [31] for speeding up DTW calculations. We note again that we chose to use DTW as the recognizer since the community has opted to use that for several systems [1, 2, 12, 30, 37, 48, 49].

Implementation of DTW. Dynamic Time Warping [24] is used to measure the similarity between the 1-D sequences of two gestures: $Q = q_1, q_2, ..., q_i, ..., q_n$ and $C = c_1, c_2, ..., c_i, ..., c_m$. Specifically, we construct a $n - by - m$ distance matrix D with an element $d_{ij} = ||q_i - c_j||$. DTW finds the non-decreasing path in D, starting from d_{11} and ending at d_{mn}, which has the minimum total value along this path. The left part of Figure 2 shows an example of the dynamic matching path between Q and C. Since the classic DTW algorithm dynamically searches the target matching path in the $n - by - m$ matrix, its computation cost is $O(nm)$.

Sakoe-Chiba Band. DTW needs considerable computation. We can improve on this with the Sakoe-Chiba band [31]. This is a known global constraint region for calculating the DTW similarity between two sequences. The idea of the band is to generate an envelope boundary for the underlying sequence. The envelope is wider when the underlying candidate sequence is changing rapidly, and narrower when the query sequence plateaus [18]. Specifically, the upper and lower bands are $U_i = max(c_{i-r} : c_{i+r}); L_i = min(c_{i-r} : c_{i+r})$. U_i, L_i, c_i is the i^{th} point of the upper, lower and candidate sequences. The lower part of Figure 2 shows an illustration of this. If another sequence Q does not fall into the Sakoe-Chiba band of C, the system determines Q is not similar to C. This allows us to shrink the search space to a more manageable size. A key issue is the proper selection of r, the number of points looked forward and backward of a given point. We set r equal to 10% of the sequence length, based on a review [28] of more than 500 papers on DTW.

3.3 Full Space Analysis

The full space refers to the total number of distinguishable gesture possibilities that can be differentiated by a gesture recognizer. Computing the full space for matching problems is easy, for example, there are 10^4 possible

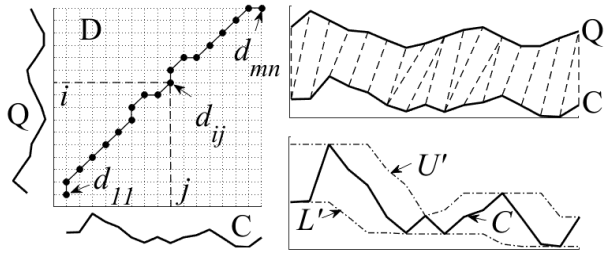

Fig. 2. An illustration of the DTW algorithm and the Sakoe-Chiba band based on one dimension of our gesture data. The left figure shows how to dynamically search along the matching path in the grid. The top right figure displays how, according to the left matching path, the points in Q map to points in C. The bottom right figure displays the Sakoe-Chiba band using the broken line; U' and L' refer to the upper and lower envelopes, respectively. The envelope is wider when sequence changes and narrower when it plateaus.

combinations of 4-digit PINs. This represents all the PINs from 0000 to 9999. However, with gestures counting is more difficult. Two gestures are technically different if even one coordinate pair is not the same. The coordinate values are real numbers with decimal places, which means that there is an extremely large, near uncountable amount of gestures that are different if you modify even one coordinate point slightly. In this section, we present the additional processing steps on raw gestures to estimate the number of all possible gestures.

3.3.1 Gesture Discretization. The key is to acknowledge that a gesture recognizer does not consider two gestures that have almost all the same points to be different. For example, the difference could be that just one x-coordinate pair differs by 0.1. A recognizer will state that the similarity of those two gestures is high, therefore it is likely they are the same gesture. The question becomes, what is the largest difference between two coordinate pairs before two gestures might be considered to be different?

A gesture coordinate pair can take on a continuous value between two sample points. This is displayed in Figure 3(a), where the red-line represents a continuous connection between the points. Gesture discretization imposes a rule on the gesture values: gesture values cannot vary continuously, they must vary by a fixed amount. If gestures vary by a fixed amount between points, then they become countable. The first step is to figure out the Discretization Level, N, which is the number of discrete values that are possible between two gesture points.

Discretization Level. We need a discrete, countable number of values between two gesture points to be able to enumerate the space. To do this, we can subdivide the range between two coordinate points into N units. This means there are N possible values between two gesture points when counting all possible gestures as opposed to the large, near-infinite number of points when the gesture remains continuous. N should be the smallest possible value while still keeping two distinct gestures – like a circle and a square – from being counted as the same gesture. To achieve a reference for N, we examine the values of N (from 5 to 40 in steps of 5) based on the EER in Dictionary dataset, as we did when determining the resampling rate R. We chose $N = 15$ units since it is a local minima for error rates. We note that $N = 15$ units is not the optimal value for any particular gesture but optimal for the dataset as a whole. Figure 3(a) shows an example of this, imposing countable units. The effect of discretization of variations is that it limits the options of samples in gesture variations to N discrete values.

Jump Limit. We have imposed a set number of values a gesture can take between sampled points when counting. However, there is a question about how large can a gesture value change? Figure 3 shows that, after gesture discretization, the adjacent discretized gesture points should not be too large. We require a semblance of continuity in order to preserve the shape of the gesture, otherwise a circle turns into a square. For example, if one sample point is at the maximum end of the value range, its adjacent sample points is unlikely to be at the

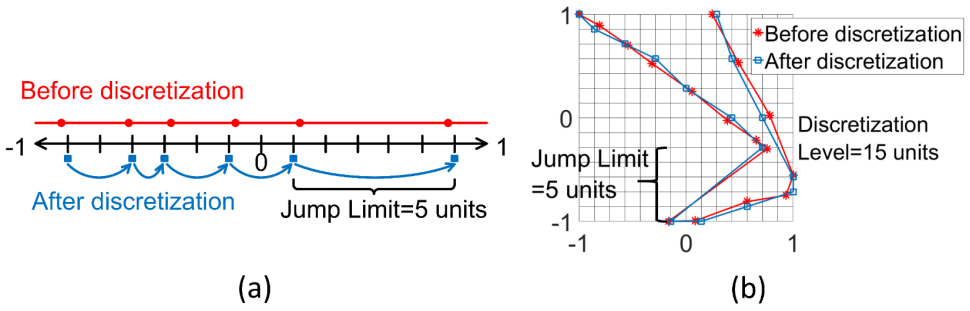

Fig. 3. Example of gesture discretization. (a) In "Before discretization" above the axis, the round points are several values of gesture samples. It can be any value between the range [-1,1]. In "After discretization" under the axis, the values of gesture samples are rounded to the closest discrete values shown as square points. (b) shows an example of a gesture under discretization. The Discretization Level is 15 units, since the range [-1,1] is divided to 15 units. The Jump Limit is 5 units, since the maximum gap between the adjacent points is 5 units.

minimum end of the value range. If this is not controlled then many details between the two points are lost and the gesture cannot be distinguished.

The Jump Limit, J defines the maximum possible difference between two adjacent gesture points. With resampling rate $R = 16$ and discretization level $N = 15$ units, we examined all the gaps between adjacent points in all the gestures in the Dictionary dataset. We found the largest gap is 12 units. Therefore, we set $J = 12$ units. The Jump Limit restricts that adjacent points of gesture variations must be less than the Jump Limit.

In summary, the size of the full gesture space is preprocessing-dependent and recognizer-independent. In the whole procedure to convert a raw gesture to a processed gesture in the full password space, the DTW recognizer is used to find reasonable reference values for the resampling rate and the discretization level. The reference values cannot be optimized based on any particular recognizers or databases. Therefore, those reference values can also be applied to other recognizers that measure the similarity between time sequences, such as, Longest Common Subsequences (LCS), Edit Distance, Cosine Distance, and Euclidean Distance.

3.3.2 Enumeration of Full Space. We introduced the procedures to transform a raw gesture to a preprocessed gesture that can be used in the recognizer and consequently as part of an authentication system. In this section, we apply the effects introduced in the preprocessing and discretization steps to a function for enumerating the full space shown in Algorithm 1.

Algorithm 1 $GNume()$ is used to enumerate the full space with a specific resampling rate, discretization level, jump limit and orientation invariance method. We enumerate gesture sequences from the last point to the first point. First, we initialize the size of the full space $FS = 0$, the length of gesture $L = R$ and the sum of visited points in X and Y, $X_{sum} = Y_{sum} = 0$. Then, we find all of the possible values for the gesture's last points, X_e and Y_e. Since the gesture is discretized, the value of a gesture sample point can only be a number from 1 to N. By combining X_e and Y_e with different values, we obtain the *EndPointSet* with size N^2, composed of (X_i, Y_i).

For each value in the *EndPointSet*, we check the orientation invariance method. If it uses method I, the Y-coordinate of a gesture's first and last points should be equal. Thus, the first value of the Y-coordinate, y_c, should be Y_i, the Y-coordinate of the last point. If it uses method II, the Y-coordinate of a gesture's first point should equal to the mean of all of the gesture's Y-coordinates, which is $R \times N/2$. Then, we use recursive function $GRecur()$ to enumerate all possible gestures with the given end point. Finally, the sum of FS is the size of the full gesture space.

Algorithm 1 Pseudocode for full space enumeration function GEnum(R, N, J, OrientationMethod). Starting with the gesture length and specific parameters, this code can enumerate all possible gestures in the full space with a given length.

Input: The length of gestures in the full space, Resampling rate R. The Discretization level N, the size of discretization values. The maximum gap between two adjacent sample points in a discrete gesture, Jump limit J. The method to remove a gesture's orientation variant, method I or method II.
Output: FS = Full gesture space size.

$FS \leftarrow 0$
$L \leftarrow R$ // length of gesture
$X_{sum}, Y_{sum} \leftarrow 0$ // X, Y sums of visited points
$X_e, Y_e \leftarrow 1, 2, ..., N$ // X_e,Y_e are the coordinates of a gesture's end point.
$EndPointSet \leftarrow (X_e, Y_e)$ // Combinations of X_e,Y_e.
for $(X_i, Y_i) \in EndPointSet$ **do**
  if Using Orientation Invariance Method I **then**
    $y_c = Y_i$ // Method I
  else
    $y_c = R \times N/2$ // Method II
  end if
  $FS \leftarrow GRecur(L, X_i, Y_i, X_{sum}, Y_{sum}, y_c, N, J) + FS$
end for
return FS

Algorithm 2 $GRecur()$ is the the core recursive function in $GNume()$. The inputs of $GRecur()$ keep the same meanings as in $GNume()$. When $L > 1$, it means the whole gesture sequence is not generated yet, so we need to keep recurring the function. When L reaches 1, it means the full gesture sequences are generated. We need to check if the generated sequence is a preprocessed gesture. Based on the restriction from the orientation invariance step, the Y coordinate of the gesture's first point should be equal to either its last point (Y_i) or the mean of the gesture points ($R \times N/2$). From the location invariance step, the mean of the gesture's X- and Y-coordinates, which is X_{sum}/R and Y_{sum}/R, must stay constant and equivalent to the center of a gesture's value range ($N/2$). If the sequence fulfills the above criteria, the sequence can be regarded as a preprocessed gesture. If not, the sequence should be discarded.

3.4 Weak Subspace Analysis

The weak subspace is a subset of passwords that people are more likely to choose. If an attacker has knowledge about people's password preferences, the attacker will use this knowledge to perform more efficient dictionary attacks compared to simple brute force attacks.

In this section, with the full space of gesture passwords as the reference point, we study the weak subspace for gesture passwords. First, we discuss how to determine the "The Authentication Similarity Threshold" in our analysis. Then, we show the procedure for "Weak Subspace Size Estimation". We find different variations of gestures in "Extract Representative Gesture". To estimate the similarities of representative gestures, we need to "Restrict Gesture Searching Region" to improve the search efficiency, and use Monte Carlo method to "Estimate Weak Subspace Size". Finally, we remove the potential double-counting issue by "Check Overlapping Regions".

3.4.1 Authentication Similarity Threshold. In gesture-based authentication systems, a robust threshold is crucial for recognition performance. For example, if the threshold is set too low, illegitimate users may be able to

Algorithm 2 Pseudocode for the recursive function $GRecur(L, X_i, Y_i, X_{sum}, Y_{sum}, y_c, N, J)$ in Algorithm 1. This code enumerates the gesture in reverse order of the sequence until the first point is reached. Then, it will examine if the enumerated sequence is a preprocessed gesture by checking the rotation and location invariance.

if $L > 1$ **then**
  $S \leftarrow 0$; // Initial gesture space size
  $X_{range} \leftarrow [max(1, X_i - J), min(N, X_i + J)]$;
  $Y_{range} \leftarrow [max(1, Y_i - J), min(N, Y_i + J)]$;
  for $X_j \in X_{range}$ and $Y_j \in Y_{range}$ **do**
    $L = L - 1$;
    $X_{sum} = X_{sum} + X_j$;
    $Y_{sum} = Y_{sum} + X_j$;
    $S \leftarrow GRecur(L, X_j, Y_j, X_{sum}, Y_{sum}, y_c, N, J) + S$;
  end for
  return S;
else
  $Flag_1 \leftarrow Y == y_c$; // If Orientation is Invariant?
  $Flag_2 \leftarrow X_{sum}/R == N/2$; // If Location is Invariant in X?
  $Flag_3 \leftarrow Y_{sum}/R == N/2$; // If Location is Invariant in Y?
  if $Flag_1$ **and** $Flag_2$ **and** $Flag_3$ **then**
    return 1; // Is a preprocessed gesture
  else
    return 0; // Not a preprocessed gesture
  end if
end if

mimic authorized users. Similarly, if the threshold is set too high, authorized users may be rejected due to slight variations in repeating their gesture. Since the optimal threshold varies for each dataset, we did not select the threshold by examining the EER of our datasets as we did when determining the resampling rate and related parameters. Instead, we examined the distribution of the DTW similarity scores among gestures throughout the Dictionary dataset and selected a reasonable value which can authenticate a majority of the legitimate users.

3.4.2 Weak Subspace Size Estimation. We estimate the size of the weak subspace as follows. First, we manually classify all of the weak subspace gestures by their symbols. Then, we extract several gestures of different drawing styles for each particular symbol. For each gesture of a particular drawing style under a specific symbol, we estimate the total number of similar gestures of it that have similarities less than the authentication threshold. The sum of all total numbers of the similar gestures of each drawing style for each particular symbol will form the size of the weak subspace. Figure 4 shows the procedure. Generally, there are four steps: 1) Extract representative gestures; 2) Restrict gesture searching region; 3) Estimate weak subspace size; and 4) Check overlapping regions.

3.4.3 Extract Representative Gestures. The reason for extracting representative gestures is that gestures for one symbol are not evenly distributed. For example, given a symbolic gesture, there could be ten gestures of one user's drawing style and twenty gestures of another user's style. In our weak subspace analysis, we should treat the two drawing styles from two people equally. Thus, we merge the gestures where the DTW distance is close and extract representative gestures.

First, we manually classify the gestures in Dictionary dataset by the different symbols. Then, for gestures from a specific symbol, we cluster the gestures by K-means [19, 21] and evaluate the clustering result through

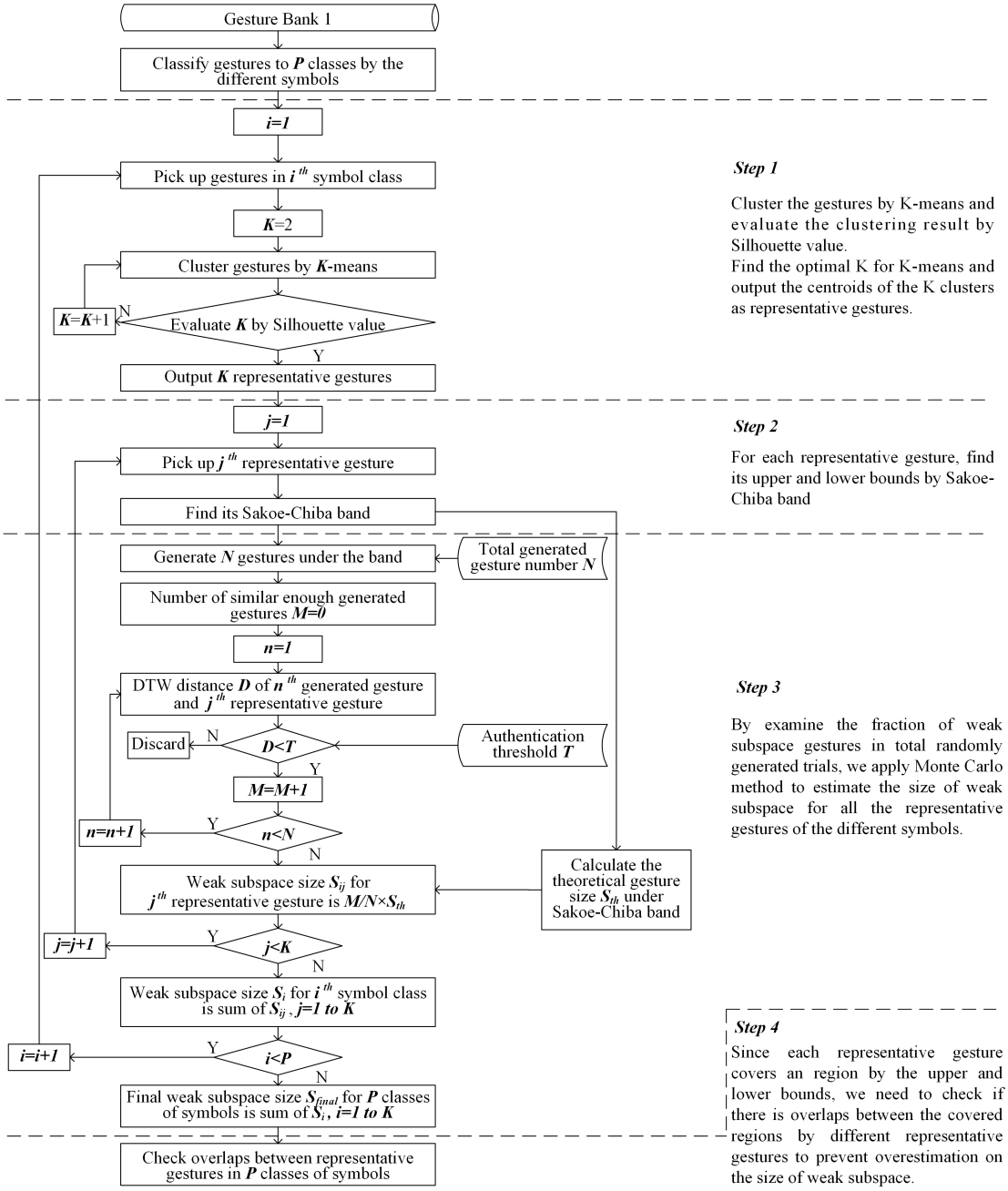

Fig. 4. The procedure to estimate sizes of the weak subspace based on data from Dictionary dataset. Step 1 is *Extract Representative Gestures*, which uses K-means and Silhouette value to extract representatives for gestures of each symbol; Step 2 is *Restrict Search Region*, which finds Sakoe-Chiba bands for each representative gesture; Step 3 is *Weak Subspace Estimation*, which is accomplished by Monte-Carlo simulation; Step 4 is *Check Overlapping Regions*, which removes potential overlaps between representative gestures' Sakoe-Chiba bands.

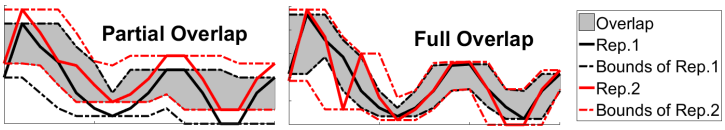

Fig. 5. Examples of two types of overlaps in weak subspace enumeration. The left is partial overlap and the right is full overlap. The grey area is the overlap area where double-counting can happen.

the Silhouette value [29]. Specifically, we start K-means at $K = 2$ and send the clustering result to be evaluated by the Silhouette. Silhouette value measures how similar a gesture is to its own cluster compared to the other clusters [29]. Thus, we can find the optimal value of K for gestures inside a specific symbol. Finally, it outputs the centroids of K gesture clusters of all gestures under one symbol as the K representative gestures.

3.4.4 Restrict Gesture Searching Region. To estimate the size of weak subspace based on representative gestures for a specific symbol, we need to find the number of similar gestures for each representative gesture.

We use the Sakoe-Chiba band of one representative gesture to restrict the searching region and find out similar gestures for the representative. With the Sakoe-Chiba band, which restricts the upper and lower band of the search region, we can calculate the size of theoretically possible gestures, S_{th}.

3.4.5 Estimate Weak Subspace Size. Even with the Sakoe-Chiba band restriction, there are still millions of possible gestures around one representative gesture. It is too expensive to compute the similarities of all of the representative gestures. Thus, we use a Monte Carlo method as an alternative estimation approach for this situation. First, we randomly generate N gestures within the Sakoe-Chiba band of a specific representative gesture. Specifically, we randomly generate the first gesture sample point and randomly generate the following points under the upper and lower bounds of Sakoe-Chiba band. Then, we measure the DTW distance, D, between the generated gestures and the representative gesture, and compare D to the authentication threshold T. If $D > T$, it means the gesture is not similar to the representative. If $D < T$, the gesture could be regarded as a similar gesture to the representative. We count the number M of similar gestures for the representative. Finally, we estimate the size of the weak subspace for one representative gesture by combining the ratio M/N and the theoretical size S_{th} of possible gestures within the Sakoe-Chiba band; this is $M/N \times S_{th}$.

By summation of the K representative gestures' weak subspace sizes for one specific symbol, we get the weak subspace size for a gesture symbol, S_i. Lastly, by adding the P classes of weak subspace sizes of different symbols, we calculate the final size of weak subspace S_{final} for gesture passwords.

3.4.6 Check Overlapping Regions. We use the Sakoe-Chiba band, which is an area around one gesture, to restrict the searching region for similar gestures for a single representative gesture. If the Sakoe-Chiba bands of two extracted representative gestures overlap, there is a possibility that the gestures in the overlap region could be double-counted by the two representative gestures as weak subspace gestures.

There are two types of overlaps between two representative gestures' bounds: (1) Partial Overlap, shown in the left of Figure 5, is when the bounds of $Rep.1$ and $Rep.2$ partially cross over; (2) Full Overlap, shown in the right of Figure 5, means that the bounds of $Rep.1$, fully fall into the area of the bounds of $Rep.2$.

We pairwise checked the representatives gestures' bands to examine whether there are any overlaps between them. We found that even among the representative gestures of the same symbol, there are no overlaps among the representative's bounds.

In summary, we have the estimated the size of the weak subspace for each gesture group and the total size of the weak subspace following the four steps above. Although we used DTW to measure the similarity between

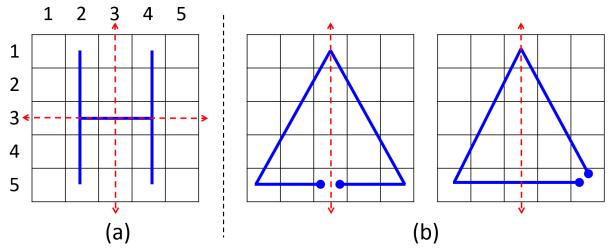

Fig. 6. Different cases for a symmetric axis. (a) A gesture symmetric by an axis in the middle of screen. It shows the two global symmetric axis in our analysis. (b) Two different locations of a symmetric axis in symmetric gestures. They illustrate that the symmetric axis does not need to be located at the center of gesture sequence.

gestures, it can be easily extended to other time sequence similarity measurements, the only two differences are the similarity measurement in K-means and the authentication threshold T. The general steps are the same.

3.5 Guessability Analysis

We present our attack method based on the dictionary of the gestures in the weak subspace. We also present a baseline brute-force attack method, which we will use for comparison.

3.5.1 Dictionary Attack Method. The order of our attack is based on the order of total weak subspace gestures in each group. The more weak subspace gestures in a group makes it more likely that the gesture group will be selected. We select a number of weak gestures in the gesture groups instead of the number in a particular gesture pattern as the index for the attack order. The reason is that since the size of the Dictionary dataset is relatively small, there is a limited number of gestures of a specific symbol meaning.

Dictionary Attack Gesture Generation Method. Generating a dictionary attack gesture uses the same process shown in Figure 4. The specific steps are:

1: The attacker selects a representative gesture in a gesture group and calculates its Sakoe-Chiba band. As a reminder, a representative gesture is extracted using K-means with a Silhouette value.
2: Randomly generate an attack gesture under the representative gesture's Sakoe-Chiba band.
3: Measure the similarity between the attack gesture and all of the target gestures under attack. If the similarity is under the authentication threshold, the corresponding gesture password is guessed correctly (cracked).
4: Repeat the above steps for another guess.

3.5.2 Benchmark: Brute Force Attack Method. Previous work has shown how, for Draw-A-Secret (DAS) graphical passwords, the weak subspace can be enumerated by using symmetric features and a small number of distinct strokes [16]. Based on prior findings from DAS [40] and Pass-Go [35] graphical passwords, we extended these ideas for our baseline brute force cracking attack. We define the following heuristics: (1) people are more likely to create symmetric horizontal or vertical axes at or near the middle of a screen; (2) people tend to choose small number of distinct strokes. The small number of distinct strokes assumed for graphical passwords is not applicable to the multistroke case for gesture passwords since we connect a multistroke gesture end-to-end in the preprocessing procedure. As a result, both unistroke and multistroke symmetric gestures should be globally symmetric. Figure 6 (a) shows an example of one type of central symmetric axis.

Brute Force Attack Gesture Generation Method.

The specific steps for the attack are as follows:

1: Randomly generate the eight X- and Y- coordinates of the first half of the gesture in the entire space.
2: Randomly select the gesture's symmetric axis: vertical or horizontal.

Table 3. The sizes of the weak subspaces for free-form gestures. "Weak Gesture #" is the number of gestures that people selected as passwords. Compared to the size of the full gesture space (about 109 bits), the weak subspace (about 82 bits) is much smaller. Furthermore, the Geometric Shape group occupies the largest part of the total weak subspace size while Digits group has the smallest weak subspace size.

Group Name	Weak Gesture #	Orientation Invariant (Bits) Method I	Method II
Digit	37	67.1	67.7
Geometric Shape	181	82.0	81.4
Letter	114	75.3	77.3
Math Function	37	77.0	76.8
Math Symbol	18	71.6	74.6
Music Symbol	11	77.1	78.4
Special Char.	9	75.4	72.4
Total	407	82.1	81.7

3: With the half gesture and symmetric axis, generate the attack gesture sequence.
4: Relocate the symmetric axis along the gesture sequence. Randomly select a starting point in the gesture, then cut the gesture at the new starting point and concatenate the gesture's old end points.
5: Measure the similarity between the attacking gestures and all of the gestures under attack. If the similarity is under the authentication threshold, the corresponding gesture password is cracked.

Figure 6(b) shows that the symmetric axis does not need to be located at exactly the middle point of the gesture sequence. Although a symmetric axis in the rightmost of Figure 6 is not located at the midpoint of the gesture sequence, its gesture shape is still symmetric and it still should be considered as a symmetric gesture. As a reminder, we resampled all gestures to 16 points in preprocessing so the attacking method only needs to generate 16 points.

4 RESULTS

In this section, we first present the sizes of full gesture space and weak subspaces. Then, we show the cracking results of the dictionary attack and the brute force attack on Test dataset (General) and Test dataset (Weak).

4.1 Sizes of Full Space and Weak Subspace of Gesture Passwords

By running Algorithm 1, we computed the size of the full theoretical gesture password spaces: 109.0 bits for *orientation method I* and 109.1 bits for *orientation method II*. Since the difference between the full space sizes for the two orientation methods is small, 109.0 bits vs. 109.1 bits, we conclude the variability in choosing either orientation method in gesture has little effect on the size of the space.

Table 3 shows the size of the weak subspace for two different orientation invariance methods based on our group clustering. The sizes of the weak subspace (both about 82 bits) are much smaller than the full space (both about 109 bits).

We found that the Geometric Shape group has the largest weak subspace and the Digit group has the smallest weak subspace. The larger the size of the subspace, the harder it is to crack it. We can predict that passwords in the Digits group are the easiest to crack while the passwords in the Geometric Shape group are the most difficult to crack among the seven groups.

By comparing the weak subspace size between the two orientation invariance methods, we found that the difference between the methods has a limited effect on the size of the weak subspace.

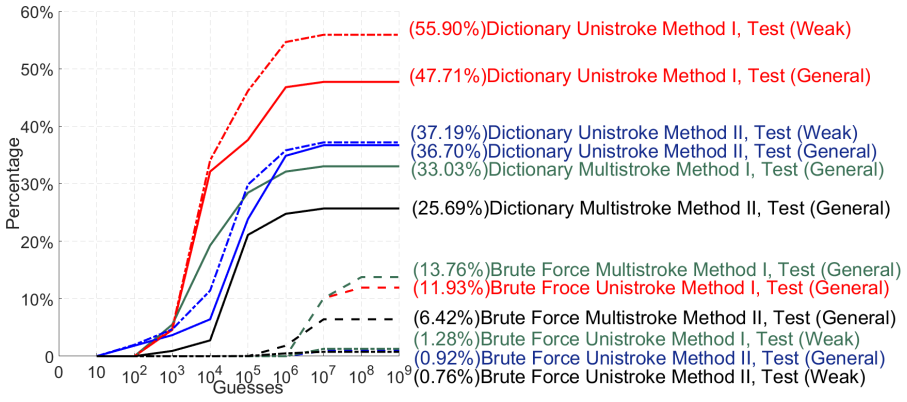

Fig. 7. Cracking results for the dictionary and brute force attacks against free-form gestures (Test (General)) and specifically collected weak subspace gestures (Test (Weak)). For Test (General), by comparing the cracking rates, we find that our dictionary attack is more efficient cracking gestures compared to a brute force attack. The orientation invariance method I is more vulnerable to attacks than method II. For Test (Weak), a comparison between our dictionary and brute force attacks is meaningless since the dataset is specifically targeted for testing the dictionary set of weak subspace gestures.

4.2 Cracking Evaluation on Free-Form Gestures

To examine the performance of the dictionary attack and the brute force attack, we perform both attacks on gestures in Test dataset (General), which consists of free-form unistroke and multistroke gestures that users selected as their passwords. Figure 7 shows the cracking results of the two attacks based on the two orientation invariance methods (marked as Test (General)).

4.2.1 Between Different Attacking Methods. Figure 7 shows that our dictionary attack is more efficient than the brute force attack. With both orientation methods, the dictionary attack cracked 35.78 percentage points more unistroke gestures and 19.27 percentage points more multistroke gestures than the brute-force attack. It is interesting that the difference between the two orientation methods does not influence different cracking rates between the attacking methods. By inspection, there does not appear to be a correlation between the gestures cracked under either orientation method and the the type of attack method.

The dictionary attack targets gestures inside the weak subspace, while the brute-force attack targets gestures with symmetric features. By checking the gestures with the particularly targeted features alone, the dictionary attack is more efficient at cracking weak subspace gestures compared to the brute-force attack cracking symmetric gestures. Figure 7 shows that the dictionary attack cracked 47.71% of unistroke gestures, which means 52 out of 109 gestures were cracked. Table 1 shows there are totally 78 weak subspace unistroke gestures in the Test dataset (General), so we find that 66.67% of weak subspace unistroke gestures were cracked. Similarly, the brute-force attack cracked 11.93% of unistroke gestures, that is 13 out of 109 unistroke gestures. Since there are 36 unistroke symmetric gestures in Test dataset (General), it means 36.11% of unistroke symmetric gestures were cracked. Therefore, the dictionary attack is more efficient at cracking weak subspace gestures than the brute force attack at cracking symmetric gestures.

4.2.2 Between Different Orientation Invariance Methods. Figure 7 shows that gestures with orientation invariance method II resists attacks better than method I. Both dictionary and brute force attacks cracked 11.01 percentage points more unistroke gestures and 7.34 percentage points more multistroke gestures with orientation method I than method II. It happens again that the difference between the two attacking methods does not

influence the cracking rates between two orientation methods. By checking the cracked gestures, we still did not find any relationships between the orientation methods and attack methods. To conclude, the selection of the orientation invariance method has an effect on the gesture's ability to resist attacks.

4.2.3 Between Different Gesture Types. As Figure 7 shows, multistroke gestures resist dictionary attacks better than unistroke gestures. With both orientation methods, there are more than 10 percentage points of unistroke gestures that were cracked than multistroke gestures.

Under brute-force attack, there is no obvious difference between cracking rates for unistroke and multistroke gestures. Additionally, the absolute numbers of cracked unistroke and multistroke gestures are small. We cannot conclude that multistroke gestures are weaker than unistroke gesture on resisting brute-force attacks.

4.2.4 Note on Computational Limitations. The computational cost of measuring the similarity between gestures is a crucial limiting factor in our test. Unlike text passwords presented in previous work [6, 14, 39], where checking if a password is right or wrong is near instant, gesture passwords requires dynamically searching for the DTW distance. The computational complexity for text password matching is $O(N)$ and for gesture matching is $O(N^2)$, with N being the password length. We spent two weeks performing 10^9 attacks against 3488 gestures, that is 218 gestures in Test dataset 1 and 3270 gestures in Test dataset 2. Assuming we want to attack the same amount of text and gesture password with the same length, the gesture password attack needs to spend $3488N^2/N \approx 10^3 N$ times more computation than text passwords. However, text based password systems can use approaches such as scrypt [27] to slow down the verification process.

4.3 Crack Evaluation on Weak Subspace Gestures

The above results show that the dictionary attack outperforms the brute force attack. To examine the effectiveness of our dictionary attacks on cracking specifically weak subspace gestures, we perform the dictionary attack on Test dataset (Weak), which contains only weak subspace gestures that were intentionally collected and covered by our gesture dictionary.

Figure 7 shows the cracking result for dictionary and brute force attacks on weak subspace gestures in Test dataset (Weak) based on the two orientation invariance methods. The dictionary attack had a 55.9% cracking rate with orientation method I and 37.19% cracking rate with method II. The brute force attack had 1.28% and 0.76% cracking rate for the two orientation methods.

We observe the cracking rates using dictionary attacks in Test dataset (Weak) (55.9% and 37.19%) is higher than that rates in Test dataset (General) (47.71% and 36.70%). The reason is that Test dataset (Weak) intentionally contains only the weak subspace gestures, while Test dataset (General) contains both weak subspace gestures and other freely created gestures. This means that Test dataset (Weak) performance is by default biased towards favoring the dictionary attack. Therefore, conclusions about the performance of the brute-force attack compared to the dictionary attack should be limited to the Test dataset (General).

Figure 8 shows the details of cracked gestures in Test dataset (Weak). The Digit gesture group is the most easily cracked while the Special Character gesture group is the most resistant. As Figure 8 shows, only 21.37% of Digit gestures can resist dictionary attacks. A possible reason is that Digit group has the smallest weak subspace (about 67 bits) in Table 3. The principal reason, however, is that there is not much observed variation in the way people perform gestures in the Digit group. In contrast, 52.99% of special character group gestures are not cracked by dictionary attacks. This can be explained by its large weak subspace size (about 75 bits) and smallest number in Table 3. The larger weak subspace size makes gestures more difficult to be cracked and the smaller number of weak gestures prevents the attack dictionary from covering all possible variations of the way people draw certain gestures.

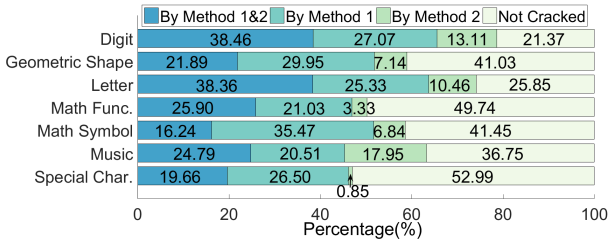

Fig. 8. Categories of cracked gestures with different orientation invariance methods in Test dataset (Weak). The Special character group is the strongest and Digit group is the weakest on resisting the dictionary attacks.

5 DISCUSSION AND CONCLUSIONS

We have presented the first paper on performing guessing attacks for gesture passwords. We developed methodology for performing guessing attacks, showed how to enumerate the size of both the full and weak subspaces for gesture passwords, identified and categorized a list of most commonly used gestures from published work on gesture passwords, and recruited participants to obtain additional data for testing our method. We showed guessing attacks against gesture passwords by creating a generalizable method based on how gestures are recognized. We extended this attack method for a dictionary attack based on common gestures as well as contributed a brute force attack based on symmetric features to use as a benchmark measurement.

Our dictionary attack method, when tested on a newly collected dataset from 109 participants, vastly outperformed the brute force attack. For unistroke gestures, this was a difference of 35.78 percentage points with Orientation Method I and a difference of 22.94 percentage points with Orientation Method II. Figure 7 shows that these types of gaps persist not matter whether unistroke, multistroke, or Orientation Method is considered: dictionary attacks are consistently better and always by double-digit percentages.

The takeaway from this is similarly clear, and it is a lesson learned in research on cracking text passwords as well: dictionary attacks informed by using the most common passwords inside of a weak subspace generates more successful attacks than a random brute-force guess. The disadvantage of the dictionary attack is also its strength: by targeting on the most common gestures it cannot crack gestures that are not covered by the weak subspace dictionary. Thus, it needs a large variety of free-form gestures to expand coverage of the weak subspace dictionary. In contrast, the brute force attack method does not need gesture samples in advance.

The size of the full space for gesture passwords is very large, despite the restrictions on the space imposed by our methodology. Table 3 shows that, given the propensity of users to select gesture passwords from the groupings we identified, the size of the full space is reduced down to 82.1 bits for Orientation Method I and 81.7 bits for Orientation Method II. Although there is a decline of over 20 bits from the full space size (109 bits) to the weak subspace size (82 bits), both the full space and weak subspace are of an impressive size. In spite of the degree of available gestures to choose from, even in an 80-bit large subspace, participants are still being routinely cracked at rates over 45%.

There is a sense that the more gestures that appear in Dictionary dataset, the larger the weak subspace size might be. As Table 2 shows, the distribution of weak set groups of Dictionary dataset and Test dataset are similar – independent of the size of each set. Currently, we computed a 82 bits weak subspace based on 407 weak gestures. Assuming we collected 100 times more gestures (40700 gestures), the 88.6 bit size would not change since we would need to assume a linear rise in the number of weak gestures that appear as well. The sizes of the weak subspaces are still much smaller than the full space size (about 109 bits), irrespective of the number of gestures. Therefore, we conclude that the larger size of Dictionary dataset may increase the size of weak subspace, but it does not have a significant influence on the relationship among the gesture groups in the weak set. The reason

is that since the relative distribution of weak set groups are constant, the analysis of the weak subspace is also constant. Even with 100 times more gestures, the size of the weak subspace cannot enlarge 100 times because the weak subspace gestures are obtained by analyzing symbols with shared meanings (Shapes, Letters). Thus, the size of our Dictionary dataset is large enough to estimate the size of the weak subspace and output reliable results.

One possibility is that the sample size is skewing the results. However, this should work in the opposite direction: since we only have low numbers of gesture passwords, and given the space is so large, it should take many more examples of gesture passwords before (ideally) seeing a collision between two users independently sampling the password distribution. Instead, given the large space, we are seeing frequent repetitions and overlap between independently and separately collected works of published gesture passwords, with our newly-collected data being cracked at a rate of 47.71%. Users are selecting gestures with meaning behind them likely because they are easier to remember and choose on-the-spot as a password. This selection bias has led to weak subspaces, however. One way to resist dictionary attacks for gesture passwords would be to advise participants to use combinations of symbols. Unlike in text passwords, even simple combinations of symbols as a gesture password should make cracking far more difficult given the large increase in computational expense. The need to address user choice for creating gestures is clear [25]. An alternative is to encourage, perhaps through policies, a methodology for generating gesture passwords that make them more random-appearing.

Each preprocessing step reduces the size of the password space. By design, that is their purpose – by reducing the total number of possibilities the intention is to make it easier for recognizers to distinguish similar gestures. This increases the likelihood of misinterpreting two gestures intended to be different as being the same. However, without preprocessing the space is much larger but the effect is that matching two gestures that are meant to be similar is far more difficult. It is easy to see why the size of the password space would be larger, though. A recognizer that does not allow location to change, for example, would distinguish two circles drawn at different corners of the screen to likely be different despite the users intentions. As a result, gestures that are too far away are considered distinct. For example, a circle that would be worth only one gesture in our preprocessed space could be worth as much as four gestures (one for each corner of the screen) in an unprocessed space. The downside is that this increases user frustrations. All of these points are in tension: the size of the password space, the recognizer efficiency, and usability of the system. Care has to be taken to balance all three.

Our experiment is conducted in the laboratory, which may affect the distribution of user-chosen gesture passwords [50]. There has not been any evidence so far of a clear difference between the distributions of laboratory and field gesture passwords [50]. As such, we cannot estimate its influence on the cracking performance.

Multistroke gestures are better than unistroke gestures at resisting cracking attempts, with the highest multistroke dictionary attack at 33.03%. The reason for this lies in the recognizer's construction of multistroke gestures. Multistroke gestures have disparate strokes connected together to form a single gesture, and these variances in drawing multiple strokes followed by connections make it more difficult to account for.

We do not examine guessing attacks or brute force attacks against gesture passwords drawn with multiple fingers (multitouch). The datasets we obtained do not have a large number of samples for these multitouch passwords. This limits our ability to make generalizations about the weak subspace groups. Previous work [34] showed biases in the distribution towards repeating unitouch figures with multiple fingers in the multitouch case. In this respect, an extension to analyze multitouch could be relatively simple. The computations are also far more expensive for the multitouch case. Multitouch gestures would be important further work to study.

In conclusion, we have presented the first paper on realistic guessing attacks against free-form gesture passwords. We believe our work is a critical step towards evaluating the security of gesture passwords. Additional material, including source code and datasets can be found at http://securegestures.org.

REFERENCES

[1] Ilhan Aslan, Andreas Uhl, Alexander Meschtscherjakov, and Manfred Tscheligi. 2014. Mid-air Authentication Gestures: An Exploration of Authentication Based on Palm and Finger Motions. In *Proceedings of the 16th International Conference on Multimodal Interaction (ICMI '14)*. ACM, New York, NY, USA, 311–318. DOI: http://dx.doi.org/10.1145/2663204.2663246

[2] Md Tanvir Islam Aumi and Sven Kratz. 2014. AirAuth: Evaluating In-air Hand Gestures for Authentication. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services (MobileHCI '14)*. ACM, New York, NY, USA, 309–318. DOI: http://dx.doi.org/10.1145/2628363.2628388

[3] Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith. 2010. Smudge Attacks on Smartphone Touch Screens. In *Proceedings of the 4th USENIX Conference on Offensive Technologies (WOOT '10)*. USENIX Association, Berkeley, CA, USA, 1–7. https://www.usenix.org/conference/woot10/smudge-attacks-smartphone-touch-screens

[4] Robert Biddle, Sonia Chiasson, and Paul C Van Oorschot. 2012. Graphical Passwords: Learning from the First Twelve Years. *Comput. Surveys* 44, 4, Article 19 (Sept. 2012), 41 pages. DOI: http://dx.doi.org/10.1145/2333112.2333114

[5] Joseph Bonneau. 2012. *Guessing human-chosen secrets*. Ph.D. Dissertation. University of Cambridge. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-819.pdf

[6] Joseph Bonneau. 2012. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy (SP '12)*. IEEE Computer Society, Washington, DC, USA, 538–552. DOI: http://dx.doi.org/10.1109/SP.2012.49

[7] Joseph Bonneau. 2014. Guessing passwords with Apple's full-device encryption. https://freedom-to-tinker.com/2014/10/08/guessing-passwords-with-apples-full-device-encryption/. (2014).

[8] Sacha Brostoff and M. Angela Sasse. 2000. *Are Passfaces More Usable Than Passwords? A Field Trial Investigation*. Springer London, London, 405–424. DOI: http://dx.doi.org/10.1007/978-1-4471-0515-2_27

[9] Sonia Chiasson, Alain Forget, Robert Biddle, and Paul C van Oorschot. 2009. User Interface Design Affects Security: Patterns in Click-based Graphical Passwords. *International Journal of Information Security* 8, 6 (Oct. 2009), 387–398. DOI: http://dx.doi.org/10.1007/s10207-009-0080-7

[10] Gradeigh D. Clark and Janne Lindqvist. 2015. Engineering Gesture-Based Authentication Systems. *IEEE Pervasive Computing* 14, 1 (Jan 2015), 18–25. DOI: http://dx.doi.org/10.1109/MPRV.2015.6

[11] Darren Davis, Fabian Monrose, and Michael K. Reiter. 2004. On User Choice in Graphical Password Schemes. In *Proceedings of the 13th Conference on USENIX Security Symposium (USENIX Security '04)*. USENIX Association, Berkeley, CA, USA, 1. https://www.usenix.org/legacy/events/sec04/tech/full_papers/davis/davis_html/usenix04.html

[12] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. 2012. Touch Me Once and I Know It's You!: Implicit Authentication Based on Touch Screen Patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 987–996. DOI: http://dx.doi.org/10.1145/2207676.2208544

[13] Alexander De Luca, Emanuel von Zezschwitz, Ngo Dieu Huong Nguyen, Max-Emanuel Maurer, Elisa Rubegni, Marcello Paolo Scipioni, and Marc Langheinrich. 2013. Back-of-device Authentication on Smartphones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2389–2398. DOI: http://dx.doi.org/10.1145/2470654.2481330

[14] Dinei Florencio and Cormac Herley. 2007. A Large-scale Study of Web Password Habits. In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*. ACM, New York, NY, USA, 657–666. DOI: http://dx.doi.org/10.1145/1242572.1242661

[15] Marian Harbach, Emanuel von Zezschwitz, Andreas Fichtner, Alexander De Luca, and Matthew Smith. 2014. It's a Hard Lock Life: A Field Study of Smartphone (Un)Locking Behavior and Risk Perception. In *Proceedings of the Tenth Symposium on Usable Privacy and Security (SOUPS '14)*. USENIX Association, Menlo Park, CA, 213–230. https://www.usenix.org/conference/soups2014/proceedings/presentation/harbach

[16] Ian Jermyn, Alain Mayer, Fabian Monrose, Michael K. Reiter, and Aviel D. Rubin. 1999. The Design and Analysis of Graphical Passwords. In *Proceedings of the 8th Conference on USENIX Security Symposium (USENIX Security '99)*. USENIX Association, Berkeley, CA, USA, 1–1. https://www.usenix.org/legacy/events/sec99/full_papers/jermyn/jermyn.pdf

[17] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio Lopez. 2012. Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy (SP '12)*. IEEE Computer Society, Washington, DC, USA, 523–537. DOI: http://dx.doi.org/10.1109/SP.2012.38

[18] Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact Indexing of Dynamic Time Warping. *Knowledge and information systems* 7, 3 (March 2005), 358–386. DOI: http://dx.doi.org/10.1007/s10115-004-0154-9

[19] Stuart P Lloyd. 2006. Least Squares Quantization in PCM. *Information Theory, IEEE Transactions on* 28, 2 (Sept. 2006), 129–137. DOI: http://dx.doi.org/10.1109/TIT.1982.1056489

[20] Alexander De Luca and Janne Lindqvist. 2015. Is secure and usable smartphone authentication asking too much? *Computer* 48, 5 (May 2015), 64–68. DOI: http://dx.doi.org/10.1109/MC.2015.134

[21] James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*. http://projecteuclid.org/euclid.bsmsp/1200512992

[22] William Melicher, Blase Ur, Sean M. Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2016. Fast, Lean, and Accurate: Modeling Password Guessability Using Neural Networks. In *Proceedings of the 25th Conference on USENIX Security Symposium (USENIX Security '16)*. USENIX Association, Austin, TX, 175–191. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/melicher

[23] Robert Morris and Ken Thompson. 1979. Password Security: A Case History. *Commun. ACM* 22, 11 (Nov. 1979), 594–597. DOI: http://dx.doi.org/10.1145/359168.359172

[24] Cory S. Myers and Lawrence R. Rabiner. 1981. A comparative study of several dynamic time-warping algorithms for connected-word recognition. *The Bell System Technical Journal* 60, 7 (Sept 1981), 1389–1409. DOI: http://dx.doi.org/10.1002/j.1538-7305.1981.tb00272.x

[25] Uran Oh and Leah Findlater. 2013. The Challenges and Potential of End-user Gesture Customization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1129–1138. DOI: http://dx.doi.org/10.1145/2470654.2466145

[26] Antti Oulasvirta, Sakari Tamminen, Virpi Roto, and Jaana Kuorelahti. 2005. Interaction in 4-second Bursts: The Fragmented Nature of Attentional Resources in Mobile HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM, New York, NY, USA, 919–928. DOI: http://dx.doi.org/10.1145/1054972.1055101

[27] Colin Percival. 2009. Stronger key derivation via sequential memory-hard functions. In *Proceedings of BSDCan '09*. Ottawa, ON, Canada.

[28] Chotirat Ann Ratanamahatana and Eamonn Keogh. 2004. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data (TDM '04)*. Citeseer, 53–63. http://wearables.cc.gatech.edu/paper_of_week/DTW_myths.pdf

[29] Peter Rousseeuw. 1987. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of computational and applied mathematics* 20, 1 (Nov. 1987), 53–65. DOI: http://dx.doi.org/10.1016/0377-0427(87)90125-7

[30] Napa Sae-Bae, Kowsar Ahmed, Katherine Isbister, and Nasir Memon. 2012. 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)*. ACM, New York, NY, USA, 977–986. DOI: http://dx.doi.org/10.1145/2207676.2208543

[31] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, 1 (Feb 1978), 43–49. DOI: http://dx.doi.org/10.1109/TASSP.1978.1163055

[32] Abdul Serwadda and Vir V. Phoha. 2013. When Kids' Toys Breach Mobile Phone Security. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS '13)*. ACM, New York, NY, USA, 599–610. DOI: http://dx.doi.org/10.1145/2508859.2516659

[33] Muhammad Shahzad, Alex X. Liu, and Arjmand Samuel. 2013. Secure Unlocking of Mobile Touch Screen Devices by Simple Gestures: You Can See It but You Can Not Do It. In *Proceedings of the 19th annual international conference on Mobile computing & networking (MobiCom '13)*. ACM, New York, NY, USA, 39–50. DOI: http://dx.doi.org/10.1145/2500423.2500434

[34] Michael Sherman, Gradeigh Clark, Yulong Yang, Shridatt Sugrim, Arttu Modig, Janne Lindqvist, Antti Oulasvirta, and Teemu Roos. 2014. User-generated Free-form Gestures for Authentication: Security and Memorability. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services (MobiSys '14)*. ACM, New York, NY, USA, 176–189. DOI: http://dx.doi.org/10.1145/2594368.2594375

[35] Hai Tao and Carlisle Adams. 2008. Pass-Go: A Proposal to Improve the Usability of Graphical Passwords. *International Journal of Network Security* 7, 2 (2008), 273–292.

[36] Julie Thorpe and Paul C van Oorschot. 2004. Graphical Dictionaries and the Memorable Space of Graphical Passwords. In *Proceedings of the 13th Conference on USENIX Security Symposium (USENIX Security '04)*. https://www.usenix.org/legacy/event/sec04/tech/thorpe.html

[37] Jing Tian, Chengzhang Qu, Wenyuan Xu, and Song Wang. 2013. KinWrite: Handwriting-Based Authentication Using Kinect. In *20th Annual Network & Distributed System Security Symposium (NDSS '13)*. http://www.internetsociety.org/sites/default/files/10_2_0.pdf

[38] Sebastian Uellenbeck, Markus Dürmuth, Christopher Wolf, and Thorsten Holz. 2013. Quantifying the Security of Graphical Passwords: The Case of Android Unlock Patterns. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS '13)*. ACM, New York, NY, USA, 161–172. DOI: http://dx.doi.org/10.1145/2508859.2516700

[39] Blase Ur, Sean M. Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L. Mazurek, William Melicher, and Richard Shay. 2015. Measuring Real-World Accuracies and Biases in Modeling Password Guessability. In *Proceedings of the 24th Conference on USENIX Security Symposium (USENIX Security '15)*. USENIX Association, Washington, D.C., 463–481. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/ur

[40] Paul C van Oorschot and Julie Thorpe. 2005. *On the security of graphical password schemes*. Technical Report. Carleton University, Ottawa, ON, USA. http://service.scs.carleton.ca/sites/default/files/tr/TR-05-11.pdf

[41] Paul C van Oorschot and Julie Thorpe. 2011. Exploiting Predictability in Click-based Graphical Passwords. *Journal of Computer Security* 19, 4 (Dec. 2011), 669–702. http://dx.doi.org/10.3233/JCS-2010-0411

[42] Emanuel von Zezschwitz, Paul Dunphy, and Alexander De Luca. 2013. Patterns in the Wild: A Field Study of the Usability of Pattern and Pin-based Authentication on Mobile Devices. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '13)*. ACM, NY, USA, 261–270. DOI: http://dx.doi.org/10.1145/2493190.2493231

[43] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. 2010. Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*. ACM, New York, NY, USA, 162–175. DOI: http://dx.doi.org/10.1145/1866307.1866327

[44] Matt Weir, Sudhir Aggarwal, Breno de Medeiros, and Bill Glodek. 2009. Password Cracking Using Probabilistic Context-Free Grammars. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy (SP '09)*. IEEE Computer Society, Washington, DC, USA, 391–405. DOI: http://dx.doi.org/10.1109/SP.2009.8

[45] Daniel Lowe Wheeler. 2016. zxcvbn: Low-Budget Password Strength Estimation. In *Proceedings of the 25th Conference on USENIX Security Symposium (USENIX Security '16)*. USENIX Association, Austin, TX, 157–173. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/wheeler

[46] Susan Wiedenbeck, Jim Waters, Jean-Camille Birget, Alex Brodskiy, and Nasir Memon. 2005. PassPoints: Design and Longitudinal Evaluation of a Graphical Password System. *International Journal of Human-Computer Studies* 63, 1-2 (July 2005), 102–127. DOI: http://dx.doi.org/10.1016/j.ijhcs.2005.04.010

[47] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07)*. ACM, New York, NY, USA, 159–168. DOI: http://dx.doi.org/10.1145/1294211.1294238

[48] Jonathan Wu, Janusz Konrad, and Prakash Ishwar. 2013. Dynamic time warping for gesture-based user identification and authentication with Kinect. In *Acoustics, Speech and Signal Processing, 2013 IEEE International Conference on (ICASSP '13)*. 2371–2375. DOI: http://dx.doi.org/10.1109/ICASSP.2013.6638079

[49] Junshuang Yang, Yanyan Li, and Mengjun Xie. 2015. MotionAuth: Motion-based authentication for wrist worn smart devices. In *Pervasive Computing and Communication Workshops, 2015 IEEE International Conference on (PerCom Workshops '15)*. 550–555. DOI: http://dx.doi.org/10.1109/PERCOMW.2015.7134097

[50] Yulong Yang, Gradeigh D. Clark, Janne Lindqvist, and Antti Oulasvirta. 2016. Free-Form Gesture Authentication in the Wild. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3722–3735. DOI: http://dx.doi.org/10.1145/2858036.2858270

Received November 2016; revised January 2017; accepted January 2017