On the In-Accuracy and Influence of Android Pattern Strength Meters

Work in Progress

Abstract—A common method for helping users select stronger authentication secrets, e.g., passwords, is to deploy a visual strength meter that provides feedback to the user while performing password selection. Recent work considered the accuracy of strength meters for passwords, but there is limited work on understanding the accuracy of strength meters for other knowledge-based authentication types, particularly Android’s graphical pattern unlock, despite there being multiple strength meters proposed for patterns in the literature. In this work, we present a preliminary analysis of the accuracy of strength meters for Android patterns, applying the same set of techniques from previous work. Using datasets of patterns collected in several user studies as a baseline, we compare strength meter estimations using weighted Spearman correlation. Overall, we find that strength estimations based on visual properties of the patterns (such as length, intersections, overlapping nodes, and similar) provide strength estimations that have low correlation with the real guessability of Android patterns. Motivated by these findings, we describe a set of research questions and experiments that are in progress that question whether the accuracy of a meter should even be the driving factor for nudging users to more secure choices.

I. INTRODUCTION

Strength meters are commonplace during password selection, and studies have shown that user choice is influenced by them [40]. Sometimes, users are forced to achieve a high meter score, especially for high-value accounts [20]. Some meter designs take into account estimations of strength based on our understanding of how users select passwords [47] or provide feedback for improving user choice [38]. However, many password meters, particularly those currently deployed across the Web, do not well relate with our current understanding of password strength [42], [9], which considers large guessing attacks where the attacker has some knowledge of likely passwords. Meanwhile, most meters are prescriptive [17], e.g., only considering if a password is of sufficient length with the appropriate number of special, upper and lower case letters, which does not correlate well with guessing strength [24]. Meter inaccuracy may even cause more harm than good by providing users with a false sense of security [41], [39].

In the mobile authentication domain, strength meters have also been proposed for Android’s graphical unlock scheme [1], [33], [31]. As a graphical knowledge-based authentication scheme, users select patterns by traversing a grid of 3x3 nodes without lifting, avoidance, or repetition (cf. Figure 1). All currently proposed pattern meters are based on visual properties, i.e., the shape of the pattern, rather than statistical features involved in the non-uniform selection of patterns by users, e.g., users tend to start in the upper left and end in the lower right. A number of studies have shown that human pattern choices are far simpler to guess and predict than selecting uniformly from the set of all possible 389, 112 patterns [37], [3], [31], [36], [15], [45].

In this work in progress paper, we analyze the strength meters proposed for Android patterns. Using a large data set of patterns collected in a number of different studies (summarized by Aviv and Dürmuth [5]), we evaluate each strength meter under realistic guessing attack conditions. The large data set, to which we refer to as “All,” consists of 4,637 patterns that are merged from four different user studies. Based on their statistical properties, we divided the dataset into three groups consisting of “Weak,” “Medium,” and “Strong” patterns. Overall, we find that meters based on visual properties greatly misrepresent the strength of a pattern and have low correlation with the real strength. In contrast, we recommend using a data-driven approach by using Markov models trained on user-chosen patterns to build an accurate meter.

However, a well-tuned strength meter must also well influence users, and based on these results, we propose a new set of studies to understand better what makes a good strength meter design. While we consider the approach recently proposed by Ur et al. [38], which tells users what is wrong with their password and how to improve it, a good starting point, we think that due to the different attack surface of patterns (namely, being a throttled attack) and the mobile setting it might not be the right solution for this setting. We also acknowledge that even the presence of any strength meter, regardless of accuracy, may positively influence users to select patterns differently. On the other hand, a more obtuse, strictly-enforced blacklisting approach without any user interaction may lead to user frustration and might not increase security [25].

Instead, we argue that a pattern strength meter does not need to be accurate as long as it will drive users away from weak and ineffective authentication choices. Motivated by prior
work in understanding perceptions of pattern strength [7], we will design a new, trained placebo meter that is tuned to best influence users. However, as the perception of security is not real security, we will also make use of a non-enforcing blacklisting approach to help users avoid selections of very common patterns. In summary, our paper consists of two parts:

(i) Completed: We implement and analyze the strength meters proposed for Android patterns and our proposal based on a Markov model. Using a large dataset of patterns collected in a number of different studies we measure the similarity between the various strength meter outputs and an ideal reference.

(ii) Work in progress: We plan to conduct a user study to evaluate a trained placebo meter that is tuned to best influence users by following users’ perception of strength. While avoiding too common patterns via a non-enforcing blacklisting approach, we like to investigate the influence of strength meter accuracy.

II. BACKGROUND AND RELATED WORK

Android unlock patterns remain a popular mobile device lock mechanism [44], [19], [26]. It was first introduced in 2008 and is based on the Pass-Go scheme [34]. While there exist some minor vendor specific variations, the patterns are typically “drawn” using 9 nodes arranged in a 3x3 grid as visualized in Figure 1. A valid pattern consists of 4 to 9 nodes, only straight lines are allowed, no node can be selected more than once, and all nodes along a path are connected (unless a node was selected before).

Android’s graphical password has been studied in many contexts, including for security (e.g., smudges [8], [13], shoulder surfing [43], [4], [18], other side-channel attacks [50], [14], [48], [49]), user choice [37], [1], [28], selection aids [36], [15], and under modifications [3], [35]. As mentioned, there has been a number of proposals for pattern strength meters [1], [33], [31], which we will describe in more detail in the following section. From this prior work on Android patterns, it is known that users tend to select non-uniformly in the space of the possible 389,112 possible patterns. There exists a common set of preferences, e.g., starting in the upper left and ending in the lower right, that make user-selected patterns highly predictable, repetitive, and easily guessed using methods such as Markov models [37]. Even then, if an attacker were to guess only the set of most common patterns, such as the Z-shaped pattern as depicted in Figure 1, the attacker would be successful in many circumstances.

These facts motivated the design of strength meters for Android patterns. However, all of these designs are based on visual features (e.g., the length, or intersections of a pattern) rather than statistical properties; as a result, the accuracy and usefulness of these meters have been questioned. Closest to our work in analyzing the accuracy of pattern meters is a poster presented by Heidt and Aviv [27] in 2016. There, the authors compared the strength reported by meters measured across all patterns to a data-driven estimation of strength based on Markov modeling. Here, we expand greatly on this effort by adopting the techniques of Golla and Dürmuth [24] in evaluating the accuracy of text-based password strength meters. These methods are described in detail in the later sections.

A. Pattern Strength Meters

We implemented three Android unlock patterns strength meters: Andriotis et al. [1], Sun et al. [33], and Song et al. [31]. Each of the pattern meters is referred to by the first author’s last name. Each of these meters is based on different formulas and slightly different visual features; each will be described in the context of the meter. For consistency, we use the term node to refer to a contact point in the grid, and the length of a pattern is defined as the total number of nodes used in its definition. When referring to specific nodes in the grid, we use the numeric labeling, starting with 0 in the upper left and ending with 8 in the lower right, as depicted in Figure 1. A pattern \( P \) is defined as a sequence of nodes, e.g., \( 0.1.2.4.6.7.8 \) for the Z-shaped pattern. We use cardinality to define the length of a pattern, like \( |P| \), which is the total number of nodes needed to draw the pattern. For example, the Z-shaped pattern is of length 7, and the T-shaped pattern of \( 7.4.1.0.2 \) is of length 5. Additionally, we use the term segment to refer to connected lines of the pattern. For example, the Z-shaped pattern has six segments, while the T-shaped pattern only has four segments.

1) Andriotis Meter: In 2014, Andriotis et al. [1] proposed an Android unlock pattern strength meter. The heuristic-driven meter is based on five visual properties: starting node, length, direction changes, knight moves, and the existence of overlapping nodes.

1A: Starting Node \( s \): \( s = 1 \) if the starting node is not the top-left corner (node 0), else \( s = 0 \).

1B: Minimum Length \( l' \): \( l' = 0 \) if the length \( l = |P| \) of the pattern is five or shorter, else \( l' = |P| - 5 \).

1C: Direction Changes \( c \): \( c = 1 \) if the number of direction changes (when three consecutive nodes are not on a straight line but form an angle different from 180 degrees) is at least 2, else \( c = 0 \). (For example, 2.4.3.5 has two direction changes.)

III. PRELIMINARIES

In this section, we present three strength meters proposed for Android unlock patterns. Additionally, we describe our attacker model and the four different datasets of patterns used in our evaluation.
1D: **Knight Moves** $k$: the number $k$ of segments with Euclidean distance $\sqrt{5}$ (e.g., 0.7, or 1.8), like the knight moves in chess.

1E: **Overlapping Nodes** $o$: the number $o$ of nodes that are retraced (e.g., the pattern 1.4.0.2 has one overlapping node, namely 1 is retraced when connecting 0 with 2).

The score $S_{Andriotis}$ (ID: 1F/1G) is then defined as:

$$S_{Andriotis} = s + l' + c + k + o$$

**Quantization (Q3)**: The score $S_{Andriotis}$ is divided into 3 strength categories: Weak ($0 \leq \text{score} \leq 1$), Medium ($\text{score} = 2$), and Strong ($\text{score} \geq 3$).

2) **Sun Meter**: Also in 2014, Sun et al. [33] proposed the following strength meter construction. Again, the meter is based on four visual properties: length, physical length, intersections, and overlaps. The authors motivated their construction by following (the nowadays outdated) NIST’s SP 800-63-2 [12] to calculate the entropy for a randomly selected password, which makes length the overall driving factor for strength estimation.

2A: **Length** $l$: is the length $|P|$ of the pattern.

2B: **Physical Length** $l'$: is the Euclidean distance of the pattern (drawn on a unit grid).

2C: **Intersections** $i$: is the number of intersections, where segments cross each other. For example, the pattern 4.0.1.3 has one intersection where 4.0 crosses 1.3. More evolved is the pattern 1.4.5.3, which has one intersection where 1.4 touches 5.3 in node 4.

2D: **Overlapping Segments** $o$: the number $o$ of segments that are retraced. For example, the pattern 1.4.0.2 has no overlapping segments, but the T-shaped pattern 7.4.1.0.2 has one overlapping segment, namely 0 which is retraced when connecting node 0 with node 2.

The score $S_{Sun}$ (ID: 2E/2F) is defined as

$$S_{Sun} = l \cdot \log(l^e + i + o)$$

**Quantization (Q5)**: The normalized score $S_{Sun}$ is divided into 5 equal intervals that correspond to the strength categories: Very Weak, Weak, Medium, Strong, and Very Strong.

3) **Song Meter**: In 2015, Song et al. [31] proposed another meter based on visual properties. In contrast to previous work, their proposal introduces weights that are adjustable and can be used to change the importance of different visual features.

The meter is defined as follows:

3A: **Length in Maximum Norm** $l^\infty$: is the length of the pattern wrt. the maximum norm (where the distance between two nodes $(x, y), (x', y')$ is $\max\{|x - x'|, |y - y'|\}$).

3B: **Intersections (Restricted)** $i$: is the number of intersections, where segments cross each other but restricted by not counting intersections that occur at the start or the end of a segment. Following this interpretation, the aforementioned pattern 1.4.5.3 has no intersections, as the “touch” happens at the end of the 1.4 segment.

3C: **Ratio of Non-Repeated Segments** $n$: is described loosely by Song et al. as “the number of times a segment does not appear in the longest repeated sub-pattern” to “the total number of segments in an input pattern p.” We understand this to imply how many uniquely directed segments (considering only the absolute direction) appear in the pattern compared to the total segments. The Z-shaped pattern has two uniquely directed segments, the horizontal (0.1; 1.2; 6.7; 7.8) and the diagonal uniquely directed segment (2.4; 4.6). Furthermore, it has six segments in total resulting in a ratio of $2/6 = 1/3$.

The score $S_{Song}$ (ID: 3D/3E) is defined as

$$S_{Song} = w_L \cdot \frac{l^\infty}{15} + w_N \cdot n + w_I \cdot \min\{i, 5\}$$

After conducting a 101 participants user study, Song et al. suggested to use the following weights: $w_L = 0.81$, $w_N = 0.04$, and $w_I = 0.15$. Sadly, their description of the meter is not complete, and leaves some room for interpretation around the “Ratio of Non-Repeated Segments,” leading to the interpretation presented above, even after multiple attempts to get more precise descriptions from the authors. Thus, our implementation of their meter follows our interpretation based on their description [31] and related work [27].

**Quantization (Q3)**: The normalized score $S_{Song}$ is divided into 3 strength categories: Weak (0.00-0.40), Medium (0.41-0.56), and Strong (0.57-1.00).

<table>
<thead>
<tr>
<th>TABLE I. ANDROID GATEKEEPER: RATE-LIMITING</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Guesses</strong></td>
</tr>
<tr>
<td>Android 6</td>
</tr>
<tr>
<td>1 guess</td>
</tr>
<tr>
<td>3 guesses</td>
</tr>
<tr>
<td>10 guesses</td>
</tr>
<tr>
<td>30 guesses</td>
</tr>
<tr>
<td>100 guesses</td>
</tr>
<tr>
<td>200 guesses</td>
</tr>
<tr>
<td>300 guesses</td>
</tr>
</tbody>
</table>

**B. Threat Model**

We consider a throttled guessing attack [10]. In such an attack, the $n$ most common secrets are guessed in decreasing order of success. It follows the idea that an attacker will always guess the most common secrets first to maximize the overall success probability of the attack. Currently, in contrast to passwords, datasets of Android unlock patterns are not easy to get. Obtaining a ranking from frequent to infrequent patterns is even more difficult. Depending on the OS version, Android implements guessing throttling which rate-limits the number of guesses an attacker can make on the device before time-outs and, eventual, lockout. The accumulated time following timeouts at various guess numbers are shown in Table I. For Android 6 (which has the highest market share of 21.6 % [32] currently), 200 guesses is a reasonable assumption for a throttled attacker, especially considering one of our datasets has 199 example patterns. For the more modern Android 7 through 9 (accumulated market share of 48.5 % [32]) we consider up to 100 guesses (10 h 45 m 30 s [2]) to be reasonable.

One instance that we evaluate requires a more advanced attacker in possession of the top 20 patterns of the respective distribution. Whether or not such an attacker is a realistic assumption is open to speculation. It is known that the pattern distribution is significantly influenced by underlying demographic factors such as age, gender, and experience in IT [28],
and potentially more such as cultural background and writing style [6]. We think a reasonably invested attacker might have access to a generic pattern dataset and its top 20 patterns, but most likely is not able to determine the exact order of the most common 20 patterns of the respective population.

C. Datasets

To measure the accuracy of the strength meter proposals (cf. Section III-A), we selected four datasets. The datasets contain patterns collected in various user studies by:

- Aviv et. al [3] (ABK-MP) in 2015
- Løge et. al [28] (LDR) in 2016
- Uellenbeck et. al [37] (UDWH) in 2013
- Von Zezschwitz et al. [45] (ZEBoDLAH) in 2016

An overview of the datasets is given in Table II. A more detailed description on how the data was collected, their statistical and visual properties, and their indented use case is given by Aviv and Dürmuth [5]. While a larger corpus of studies exist [5], only a minority shared their datasets or collected variants of the classical patterns that are not comparable. Overall, we evaluated 4,637 human-chosen Android unlock patterns.

We report the throttled guessing resistance in terms of \( \beta \)-success-rate and Min-entropy (\( H_\infty \)). The \( \beta \)-success-rate measures the expected guessing success for an adversary limited to \( \beta \)-guesses per account (\( \lambda_1 = 1 \) guess). The Min-entropy is a worst-case metric for user-chosen secrets, considering an attacker who only guesses the most likely secret before giving up. An introduction to the two metrics is given by Bonneau [9] and Wang et al. [46].

![Table II: Datasets: Guessing Difficulty](image-url)

The 4,637 patterns are merged from the four different user studies, and we will refer to them as “All” in the following sections. Based on their statistical properties we sorted the datasets from the individual studies into three groups, consisting of “Weak,” “Medium,” and “Strong” patterns, similar to what was done by Aviv and Dürmuth [5]. Due to the limited number of samples, guessing 100 patterns (in perfect order), for the two individual datasets ABK-MP-adv-defensive and UDWH-defensive results in very high \( \lambda_{100} \) values. However, the merge of the datasets consists of 199 unique samples, explaining the very different (albeit more realistic) \( \lambda_{100} \) value reported for the “Strong” dataset.

To better understand the overall level of protection offered by Android unlock patterns, we added statistics for 4-digit [11] and 6-digit [46] PINs and passwords that were created on mobile devices [29] as alternative mobile authentication schemes. For 4-digit PINs, we used Daniel Amitay’s iPhone PIN dataset [11]. As there is no real-world 6-digit PIN data available, we contacted Wang et al. [46] to obtain a copy of their artificially crafted 6-digit PIN dataset. Unfortunately, they were not allowed to share their PINs with us. However, as their dataset has been generated by extracting PINs from password leaks, they recommended us to do the same. By following their PIN extraction method [46], we extracted 4 and 6-digit PINs from the RockYou password leak [16]. The \( \beta \)-success-rate and Min-entropy numbers of our extracted data are the same as the numbers reported by Wang et al. (requires the conversion from bits to percentages). However, one must be careful in drawing conclusions from the 6-digit artificial dataset, as the extracted data may not follow the same distribution as “normal” 6-digit PINs. For example, we observed that especially “123456” (the most common PIN) is overrepresented in this generated dataset (the PIN is up to 20x more frequent) resulting in a relatively high \( \lambda_1 \) value.
Besides PINs, passwords created on mobile devices are another common knowledge-based authentication mechanism. In 2016, Melicher et al. [29] studied the strength and usability of passwords that were created on mobile devices. We report the β-success-rate and Min-entropy for two conditions they studied. While both datasets of passwords were created and used on mobile devices (MM), they differ by their password composition policy. The first, 3classSM, loosely follows the nowadays outdated NIST recommendation [12], by requiring at least 8 characters and at least three character classes. Furthermore, Melicher et al. tested a more usable and more secure alternative that requires at least 12 instead of 8 characters and also at least three character classes, called 3class12MM.

IV. Markov Models Strength Estimation

In this section, we outline how to calculate pattern strength based on Markov models, as a form of a data-driven meter.

To estimate the strength of patterns, Markov models have been used by Aviv et al. [3], Uellenbeck et al. [37], and Løge et al. [28]. Furthermore, the authors of pattern-creation guiding systems like SysPal [15] and TinPal [36] and alternative layouts like Pass-O [35] used Markov models to evaluate their proposals.

Markov models are based on the observation that subsequent tokens, such as letters in normal text or nodes in the pattern, are rarely independently chosen by humans and can often be quite accurately modeled based on a short history of tokens. For example, in English texts, the letter following a t is more likely to be an h than a q. For Android pattern unlock, a similar process occurs by which users are more likely to choose some transitions over others based on prior state, e.g., starting in the upper left, a user will more likely move to the right as opposed to the center node.

Using these observations, one can construct an n-gram Markov model that estimates the likelihood of a pattern by considering the probability of a transition from the previous $n-1$ nodes to the next node. For a given sequence of nodes for a pattern $p = \{c_1, \ldots, c_m\}$, an n-gram Markov model estimates its probability as

$$P(c_1, \ldots, c_m) = P(c_1, \ldots, c_{n-1}) \prod_{i=n}^{m} P(c_i|c_{i-n+1}, \ldots, c_{i-1}).$$

The required initial probabilities $P(c_1, \ldots, c_{n-1})$ and transition probabilities $P(c_i|c_{i-n+1}, \ldots, c_{i-1})$ can be determined empirically from the relative frequencies from training data. One commonly applies further post-processing to the raw frequencies: So-called smoothing tries to even out statistical effects in the transition matrix and initial probabilities. In particular, smoothing avoids relative frequencies of 0, 1, or any other boundary value. A suitable weight could be the frequency of a pattern, as it is a direct indicator for the “importance” or commonness of a pattern. Furthermore, the “Spearman’s rank correlation coefficient” to be a useful candidate to measure the accuracy of a strength meter compared to the ideal reference.

In the following, we try to give an intuitive explanation of this coefficient. The “weighted” part means that some patterns are more important than others. For example, misjudging common patterns as “strong” is considered very bad, as such patterns are usually guessed early in an attack. In contrast, incorrectly over- or underestimating rare patterns is considered less problematic. A suitable weight could be the frequency of a pattern, as it is a direct indicator for the “importance” or commonness of a pattern. Furthermore, the “Spearman’s rank” part means that the metric operates on the relative ranking instead of the direct meter output. Meters report strength values in different formats. Most common are home-grown scores, bits, probabilities, guess numbers, or the time required to guess the secret. To avoid any issues arising from the different output formats the metric operates on the relative ranking only. Thus, it is interested in whether pattern A is stronger than pattern B, but not by how much. Finally, the metric is a correlation coefficient thus it correlates two data points with each other.

We tested two advanced models (ID: 4B/4C): In the first (ID: 4B), we train individual Markov models per pattern length, motivated by previous work [23]. This approach typically yields better approximations and has been used before in the context of pattern security [3]. The latter variant (ID: 4C), follows the normal construction and training as described above, but additionally hard-codes the top 20 most frequent patterns of the respective datasets, i.e., “All,” “Strong,” “Medium,” or “Weak.” Guessing those 20 patterns in perfect order (without any approximation errors owed to the Markov model) improves accuracy. However, it requires an attacker that is in possession of this top 20 data, in contrast to an attacker that only trains a more generic Markov model based on all patterns available.

V. Meter Accuracy

We now present the results of measuring the accuracy of each of the strength meters (cf. Section III-A) and our proposed Markov model-based meter approach (cf. Section IV). From that, under our attacker model (cf. Section III-B), we show how the meters that rely on visual features inaccurately reflect the real strength of patterns.

A. Measuring Strength Meter Accuracy

The accuracy is a vital component for the overall performance of a strength meter, particularly for text-based passwords. Recent work by Golla and Dürmuth [24] systematically studied what constitutes a fair comparison of strength meters. The preferred method to measure the accuracy of a strength meter is to compare it to an ideal reference, measuring the similarity between the reference and the meter output. In their work, the authors suggest the weighted Spearman’s rank correlation coefficient to be a useful candidate to measure the accuracy of a strength meter compared to the ideal reference.

In the following, we try to give an intuitive explanation of this coefficient. The “weighted” part means that some patterns are more important than others. For example, misjudging common patterns as “strong” is considered very bad, as such patterns are usually guessed early in an attack. In contrast, incorrectly over- or underestimating rare patterns is considered less problematic. A suitable weight could be the frequency of a pattern, as it is a direct indicator for the “importance” or commonness of a pattern. Furthermore, the “Spearman’s rank” part means that the metric operates on the relative ranking instead of the direct meter output. Meters report strength values in different formats. Most common are home-grown scores, bits, probabilities, guess numbers, or the time required to guess the secret. To avoid any issues arising from the different output formats the metric operates on the relative ranking only. Thus, it is interested in whether pattern A is stronger than pattern B, but not by how much. Finally, the metric is a correlation coefficient thus it correlates two data points with each other.
In our case, the ground truth (sometimes called reference) is correlated with the meter output. We use the frequency (i.e., how often a pattern occurs) in a given dataset as the reference. Using the frequency, thus how likely a secret is, is the standard metric in throttled guessing scenarios. It follows the idea that an attacker will always guess in decreasing order of success, thus will guess very likely/common patterns first.

**Implementation:** All strength meters are implemented using the Kotlin programming language. This allows to run them natively on Android devices and enables us to transpose it to JavaScript to run them in a Web browser, too. The accuracy evaluation is implemented using R v3.5.1 (July 2018). For the weighted Spearman correlation we use the **wCorr** package. All meter implementations and evaluation scripts are available online [30].

### B. Results

An overview of the results is given in Table III. If the reported weighted Spearman correlation is close to 1.0 (high positive correlation), the meter accurately estimates pattern strength. If the correlation is close to 0.0 (no correlation), the meter is randomly guessing and does not estimate strength. Each meter has the potential to guide users in the wrong direction or confuse them. If the correlation is close to –1.0 (high negative correlation), the meter is accurate, but “strong” patterns are in fact “weak” and the other way round, misleading users into choosing weak patterns instead of strong. In cases where the evaluated feature does not occur within the 200 most frequent patterns of the respective dataset, we display a hyphen (–).

In the following, we report the accuracy of various strength meters and their considered features by providing the total range from smallest to highest correlation value “(smallest-highest)” observed across all four datasets. Unsurprisingly all meters that rely on visual features fail to estimate strength accurately, given a throttled guessing scenario. Most promising are the visual features:

- **1A Starting Node (0.160-0.405),**
- **1C Direction Changes (0.249-0.406).**

Thus, especially the ID: 1F Andriotis meter that relies on both of those features performs most accurate among the meters considering only visual features. For the ID: 2E Sun meter we find results close to zero (−0.030-0.051). ID: 3D Song meter performs similarly inaccurate (−0.050-0.034). We found that both, Sun’s as well as Song’s meter do not estimate strength accurately in the considered throttled guessing scenario.

This result is not completely surprising, as similar observations have been made before in the context of LUDS-based password strength meters, counting the different character classes. A more intuitive explanation, about what is wrong with the meters, could be the following: Considering the aforementioned visually very complex pattern in Z-shape (0.1, 2, 4, 6, 7, 8). The pattern is the second most common pattern in the datasets, thus has a guess number of 2.

<table>
<thead>
<tr>
<th>ID Meter Quant.</th>
<th>Throttled Attacker - Top 200</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
</tr>
<tr>
<td>1A Starting Nodes</td>
<td>0.395</td>
</tr>
<tr>
<td>1B Minimum Length</td>
<td>−0.034</td>
</tr>
<tr>
<td>1C Direction Changes</td>
<td>0.335</td>
</tr>
<tr>
<td>1D Knight Moves</td>
<td>0.139</td>
</tr>
<tr>
<td>1E Overlapping Nodes</td>
<td>0.277</td>
</tr>
<tr>
<td>1F Andriotis Q3</td>
<td>0.121</td>
</tr>
<tr>
<td>2A Length</td>
<td>−0.085</td>
</tr>
<tr>
<td>2B Physical Length</td>
<td>−0.024</td>
</tr>
<tr>
<td>2C Intersections</td>
<td>0.161</td>
</tr>
<tr>
<td>2D Overlapping Segments</td>
<td>0.067</td>
</tr>
<tr>
<td>2E Song</td>
<td>−0.030</td>
</tr>
<tr>
<td>2F Sun Q5</td>
<td>−0.056</td>
</tr>
<tr>
<td>3A Length (Max. Norm)</td>
<td>−0.075</td>
</tr>
<tr>
<td>3B Intersections (Rest.)</td>
<td>0.090</td>
</tr>
<tr>
<td>3C Ratio NR Segments</td>
<td>0.166</td>
</tr>
<tr>
<td>3D Song</td>
<td>−0.050</td>
</tr>
<tr>
<td>3E Song Q3</td>
<td>0.094</td>
</tr>
<tr>
<td>4A Markov (2-grams)</td>
<td>0.313</td>
</tr>
<tr>
<td>4B Markov (3-grams)</td>
<td>0.627</td>
</tr>
<tr>
<td>4C Markov (4-grams)</td>
<td>0.684</td>
</tr>
<tr>
<td>4A Markov (5-grams)</td>
<td>0.742</td>
</tr>
<tr>
<td>4B Markov (3-grams - Length)</td>
<td>0.843</td>
</tr>
<tr>
<td>4C Markov (3-grams - Top 20)</td>
<td>0.880</td>
</tr>
</tbody>
</table>

**Quantization:** Q3-Q5 = Number of bins, e.g., Q3 = [Weak, Medium, Strong].

**Weighted Spearman correlation:** 1.0 (high positive correlation); the meter accurately estimates pattern strength. 0.0 (no correlation); the meter is randomly guessing, and does not estimate strength. -1.0 (high negative correlation); the meter is accurate but “strong” patterns are in fact “weak” and the other way round.

In contrast, our results show the potential of using probabilistic approaches like Markov models to estimate the strength of a pattern. Our suggested generic Markov model (ID: 4A) reproduces the pattern ranking of the reference quite well (0.176-0.751). While one can observe an increasing accuracy with increasing n-gram size (2-grams: 0.176-0.396, 5-grams: 0.415-0.751), one needs to consider the sparse data situation and the risk of overfitting. Like previous work, we find the approach to train a separate model per pattern length (ID: 4B) to improve the accuracy (3-grams: 0.396-0.843). However, the performance degrades if not enough samples per length are available, as one can observe for the “Strong” dataset. As expected, hard-coding the top 20 patterns (ID: 4C) performs best (3-grams: 0.655-0.916) but requires a stronger attacker that is in possession of the top 20 patterns for each evaluated dataset.

The adverse effects of quantization, i.e., decreasing accuracy once the meter output is binned, for example into 3 bins (Q3) [Weak, Medium, Strong] has been found before [24]. The lower the number of bins, the more information in the relative ranking is lost; thus the weighted Spearman correlation degrades. This effect can be observed across all binning meters (ID: 1G, 2F, 3E).

---

VI. WORK IN PROGRESS: METER INFLUENCE

Next, we outline our current effort towards understanding the influence of strength meters on users’ pattern choices.

A. Motivation

Considering the stringent rate-limiting (cf. Table I) deployed in modern Android versions 7+, and a reasonably determined throttled attacker, there is no need for patterns to repel much more than 100 guesses. Forcing users to create very strong patterns above this threshold, will lead to patterns falling into the so-called “don’t care” region, as introduced by Florêncio et al. [21] in the context of password strength. Even worse, in contrast to passwords, there is no chance for a pattern to survive an unthrottled guessing attack, due to the small theoretical keyspace. Generally speaking, every pattern in this “don’t care” region must be considered a waste of cognitive effort.

All three proposed pattern meters were evaluated in user studies [1], [33], [31]. For example, Sun et al. tested their meter with 81 participants and found: “...the presence of a visual indicator of pattern strength did encourage users to create visually complex patterns.” It seems that the mere presence of a strength indicator that follows the intuitive thought process is enough to motivate users. We can imagine that as long as the meter follows users’ perception of strength [39], [41], it might be enough to nudge them toward choices outside the easily guessed threshold, but below the “don’t care” chasm.

Based on this scenario we claim that while a high accuracy is important for meter developers, it certainly is not the driving factor for users. Instead, we like to explore whether a meter should engage more with users’ perception of strength while still preventing too common, thus, very dangerous pattern choices. To this point, there is little known about user perception of pattern security. One study investigating this issue was conducted by Aviv and Fichter [7] in 2014. They identified the length (Euclidean distance) as one of the most influential visual features in user perception of pattern strength.

Based on those observations we developed the following questions that we like to answer by conducting a user study:

**RQ1:** How important is the accuracy of a strength meter to nudge users towards more secure patterns?

**RQ2:** Can a new strength meter that follows perceived instead of actual pattern strength, actively reduce the number of too common patterns, while not wasting users’ cognitive effort in the “don’t care” region?

B. Methodology

Our prospective study design follows previous work by Aviv and Fichter [7] and Ur et al. [39]. In the study, users are presented with an animation that shows the creation of a pattern, as one can see in Figure 2. As patterns can be very complex and hard to follow given a static image, we use CSS animations to visualize the pattern creation process and provide a button to replay this animation. In contrast to previous work, we display two interactive strength meters below the pattern. Users are then asked to judge the strength of the pattern by deciding which of the two strength meter bars is more accurate.

The two meters displayed in Figure 2, are very different. The first meter (left) is an accurate estimation of strength based on our Markov model-based proposal. The second meter (right) follows the intuitive idea of pattern strength of users based on the results from previous work [7] (which is mainly driven by visual complexity) but prevents too common choices like the Z-shape (0.1.2.4.6.7.8) pattern.

The idea of using this approach is to understand what is more influential for users: the highly accurate meter, or a trained placebo meter that behaves intuitively for users but prevents dangerous choices. After deciding which meter is more accurate, we will ask users about why they believe one meter gives more accurate feedback than the other. This way, we hope to learn more about their decision process that could help us to determine the importance of actual accuracy over perceived pattern strength accuracy.

To prevent users from choosing too weak patterns, we imagine a dialog (cf. Figure 3) similar to Apple iOS’s Passcode warning that implements a non-enforcing blacklist [22].

![](image1.png)

**Fig. 1.** Comparison of an accurate (left) and intuitive strength estimation (right). Participants need to judge which of the two strength estimates is more accurate. The pattern 1.2.5.8.7.6.3.0.4 is a common pattern, but typically outside the reach of throttled guessing attacks. Thus, the trained placebo meter should estimate it (based on its visual complexity) as “Strong.”

![](image2.png)

**Fig. 2.** Imagined warning dialog that tries to encourage users to reconsider their pattern choice, if a too common pattern is chosen. The dialog is based on Apple iOS’s Passcode warning that implements a non-enforcing blacklist [22].
We believe that the approach of using a trained placebo meter is more intuitive than strength estimations and will become easier to comprehend for users. At the same time, the design promises the same security by preventing too common patterns from being easier to comprehend for users. Yet, the meter is more intuitive than strength estimations and will instead of actual strength while maintaining security via a of a strength meter that follows perceived pattern strength to actively reduce the number of patterns that are at risk in a throttled guessing attack while not wasting cognitive user effort.

To this end, we are in the process of designing a future study to investigate how vital meter accuracy is to nudge users towards more secure patterns. We will explore the use of a strength meter that follows perceived pattern strength instead of actual strength while maintaining security via a non-enforcing blacklist of too common patterns. It is our goal to actively reduce the number of patterns that are at risk in a throttled guessing attack while not wasting cognitive user effort.

VII. CONCLUSION

In this work, we explored the accuracy of pattern strength meters using multiple datasets. Our results indicate that current proposals from the literature, which are solely based on visual features, do not estimate pattern strength and are a potentially dangerous substitute for statistical estimators, such as those based on Markov models. However, we acknowledge that the sheer presence of a meter, regardless of accuracy, is likely beneficial in the throttled guessing attacker model in which Android patterns exist. The primary goal should be to ensure that users do not select very common patterns, but are also not overburdened with attempting to select overly complex patterns.

To this end, we are in the process of designing a future study to investigate how vital meter accuracy is to nudge users towards more secure patterns. We will explore the use of a strength meter that follows perceived pattern strength instead of actual strength while maintaining security via a non-enforcing blacklist of too common patterns. It is our goal to actively reduce the number of patterns that are at risk in a throttled guessing attack while not wasting cognitive user effort.

REFERENCES


