On the Accuracy of Password Strength Meters

Maximilian Golla
Ruhr University Bochum
Bochum, Germany
maximilian.golla@rub.de

Markus Dürmuth
Ruhr University Bochum
Bochum, Germany
markus.duermuth@rub.de

ABSTRACT

Password strength meters are an important tool to help users choose secure passwords. Strength meters can only then provide reasonable guidance when they are accurate, i. e., their score correctly reflect password strength. A strength meter with low accuracy may do more harm than good and guide the user to choose passwords with a high score but low actual security. While a substantial number of different strength meters is proposed in the literature and deployed in practice, we are lacking a clear picture of which strength meters provide high accuracy, and thus are most helpful for guiding users. Furthermore, we lack a clear understanding of how to compare accuracies of strength meters.

In this work, (i) we propose a set of properties that a strength meter needs to fulfill to be considered to have high accuracy, (ii) we use these properties to select a suitable measure that can determine the accuracy of strength meters, and (iii) we use the selected measure to compare a wide range of strength meters proposed in the academic literature, provided by password managers, operating systems, and those used on websites. We expect our work to be helpful in the selection of good password strength meters by service operators, and to aid the further development of improved strength meters.

CCS CONCEPTS

• **Security and privacy → Authentication**; *Usability in security and privacy*; *Web application security*;

KEYWORDS

Strength Meter; Password; User Authentication

ACM Reference Format:
Maximilian Golla and Markus Dürmuth. 2018. On the Accuracy of Password Strength Meters. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3243734.3243769

1 INTRODUCTION

Password-based authentication is still in widespread use, specifically for online authentication on the Internet and for hard-disk encryption. Passwords are easy to understand for laypersons, easy to implement for the operator, don't require additional hardware, and are supported by a broad ecosystem such as password managers. In all likelihood, password-based authentication will stay for the foreseeable future.

Password strength meters (PSMs) are designed to help with one of the central problems of passwords, namely weak user-chosen passwords. From leaked password lists we learn that up to 20 % of passwords are covered by a list of only 5 000 common passwords [63]. A PSM (also called strength meter, password checker, or similar) displays an estimation of the strength of a password when chosen by the user, and either helps or forces the user to pick passwords that are strong enough to provide an acceptable level of security.

The accuracy with which a PSM measures the actual strength of passwords is crucial; as people are known to be influenced by PSMs [61] (or even forced to comply), an inaccurate PSM can do more harm than good. If weak passwords are rated strong users might end up choosing this password, actually harming security; similarly, if strong passwords are rated weak the meter drives away people from those strong passwords. Traditionally, ad-hoc approaches such as counts of lower- and uppercase characters, digits, and symbols (LUDS) have been used to measure the strength of passwords. Despite being well-known that these do not accurately capture password strength [68], they are still used in practice. More recently, more sound constructions for PSMs based on precise models capturing user choice have been proposed, e. g., based on Markov models [13], based on probabilistic context-free grammars (PCFGs) [34, 65], neural networks [46, 59], and others [71].

Surprisingly, very little work has been performed on a fair comparison of these different proposals, and it remains unclear which password meter is best suited for the task of estimating password strength. Even worse, we lack consensus on how to determine the accuracy of strength meters, with different techniques used ranging from Spearman and Kendall correlation to ad-hoc measures.

In this work, we propose a sound methodology for measuring the accuracy of PSMs, based on a clear set of requirements and careful selection of a metric, and we will use this metric to compare a variety of different meters. In more detail, our contributions are:

(i) We discuss properties an accurate strength meter needs to fulfill, and create a number of test cases from these requirements.

(ii) We report tests of 19 candidate measures (and have tested several more) from a wide range of types and select good metrics for the accuracy of strength meters.

(iii) We address the challenge of estimating the accuracy from limited datasets and show that meters can be reasonably approximated with a small number of random samples.

(iv) We provide an extensive overview of the current state of the art of strength meters.

CCS '18, October 15–19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5693-0/18/10…$15.00
https://doi.org/10.1145/3243734.3243769

(v) We use the derived measures and provide a comparison of a broad selection of 45 password meters in 81 variations, ranging from academic proposals over meters deployed in password managers and operating systems to meters in practical use on websites.

More important than the results of this work, are the methods we developed. They provide means to select suitable similarity metrics that match the requirements for specific use cases. We hope that it will foster the future development of strength meters and will simplify the selection process for service operators.

2 RELATED WORK

In the following, we review some material related to password strength including password choice, password guessability, and metrics that were proposed for measuring strength.

2.1 Password Choice

Jakobsson and Dhiman [37] found that users produce passwords using only a small set of rules and components such as dictionary words, replacement strategies, and misspellings. Veras et al. [64] explored the semantics of passwords. They found male/female names and concepts relating to love, profanity, animals, food, and money. Recently, Ur et al. [60] investigated the relationship between users' perceptions of the strength of specific passwords and their actual strength. They report serious misconceptions on the consequences of constructing passwords by using common phrases and including digits and keyboard patterns. A possible countermeasure to prevent users from choosing easy to guess passwords are so-called password composition policies (e. g., requiring the password to contain a digit or symbol). Komanduri et al. [42] analyzed the effect of such policies by investigating their impact on password strength, user behavior, and user sentiment. Based on their findings, they produced recommendations for password-composition policies that result in stronger passwords without burdening users too much. In 2014, Florêncio et al. [24] highlighted the importance to limit the number of online guesses that can be made by rate-limiting and blacklisting the most common passwords like "123456". Recently, Habib et al. [33] analyzed how users react to password creation attempts failing because of blacklists. They conclude that blacklist checks need to go beyond exact comparisons and also verify for any form of reuse of blacklisted passwords. Furthermore, they recommend providing textual feedback to help users understand reuse and simple modifications of blacklisted password attempts.

2.2 Password Guessing

Guessing passwords is in many ways related to password strength. Assuming an attacker cannot just invert a password hash, the optimal strategy to test passwords is in decreasing order of likelihood, i. e., most frequent passwords first. There are different proposals to enumerate passwords with decreasing likelihood, in other words, with increasing strength. The most relevant for GPU-based password cracking use large dictionaries and ad-hoc mangling rules to generate password candidates. Narayanan and Shmatikov [48] proposed Markov-models to overcome some of the problems of dictionary-based attacks. Dürmuth et al. [21] improved the approach by generating password candidates according to their occurrence probabilities, i. e., by outputting the most likely passwords first. Weir et al. [70] suggested a method exploiting structural patterns from a password leak using probabilistic context-free grammars (PCFGs). Veras et al. [64] extended the approach by building a semantically meaningful PCFG-based password guesser. An empirical study on the effectiveness of different attacks was done by Dell'Amico et al. [19]. Another study targeting probabilistic password modeling approaches was done Ma et al. [44]. Recently, Ur et al. [62] did a large scale comparison and found running a single guessing algorithm, often yields a very poor estimate of password strength. In 2016, Melicher et al. [46] proposed using recurrent neural networks (RNNs) for probabilistic password modeling.

2.3 Password Strength

Trying to estimate password strength as a measure to defend against guessing attacks has a long history. In 1979, Morris and Thompson [47] did password checking by attempting to crack hashed passwords. The ones successfully cracked were marked as weak, and the users were notified. In the following, one started to check the strength of a password before it is accepted by a system via *proactive password checkers* or *password strength meters*, using certain rules-sets that try to exclude weak passwords [7, 41, 56]. Schechter et al. [53] classified passwords as weak by counting the number of times a certain password is present in the password database. Bonneau [8] proposed α-guesswork as a proper metric for estimating the guessing entropy of a fraction α of accounts. Kelley et al. [40] proposed a *guess-number calculator* to determine if and when a given password-guessing algorithm, would guess a specific password. Dell'Amico and Filippone [18] proposed a method to estimate the number of guesses required to find a password.

Many of the password strength meters in the current literature are based on the aforementioned password guessing approaches: using neural networks by Melicher et al. [46] and Ur et al. [59], using PCFGs by Houshmand and Aggarwal [34] and Wang et al. [65], and using Markov models by Castelluccia et al. [13]. Furthermore, there is a meter using a set of advanced heuristics by Wheeler [71], the official NIST entropy estimation [11], and others [22, 32, 61]. We provided a description of the meters in Section 6.3. Ur et al. [61] found that strength meters, depending on the visual feedback, led users to create longer passwords or caused them to place less importance on satisfying the meter. Egelman et al. [22] studied the impact of password strength meters on the password selection process and found that meters result in stronger passwords when users are forced to change existing passwords on "important" accounts. de Carné de Carnavalet and Mannan [16] conducted an analysis of deployed strength meters in 2014. They found evidence that the commonly used meters are highly inconsistent and fail to provide coherent feedback.

3 PASSWORD STRENGTH METERS

In this section, we discuss password strength meters and how to measure their accuracy.

3.1 Approximating Strength

"Weak" passwords such as `passw0rd` or `abc123` are not insecure *per se* (e. g., based on some "magical" property they fulfill). They are insecure as they are chosen commonly by humans, and thus

an adversary trying to guess passwords will guess those common passwords early in an attack. (Similar observations have been made recently by Wang et al. [65].)

An *ideal strength meter*, thus, assigns each password its likelihood, e. g., approximated by the relative frequency from a large enough password corpus. However, this straightforward idea is hard to use in practice: The relative frequencies can in principle be accurately approximated for "relatively likely" passwords (cf. [8]), e. g., those that are particularly relevant for online guessing attacks. Estimating frequencies for less likely passwords, relevant for offline guessing attacks, is next to impossible due to the amount of data required. Therefore, practical strength meters should aim at approximating the true strength using compactly representable functions. The traditional LUDS meter allows for a very compact representation (of a few bytes), at the cost of limited accuracy [68], while other approaches based on Markov models [13] or PCFGs [65, 69] have been demonstrated to be more accurate, at the expense of increased storage size.

For the remainder of this work we assume a PSM is a mechanism f that takes as input a password, i. e., a string of characters Σ^n over an alphabet Σ, and outputs a score or strength value: $f\colon \Sigma^* \to \mathbb{R}$. We assume the score being a real-valued number. Some meters aim at providing an estimate for the *probability* of a password (e. g., [13, 34, 65]), i. e., values are in the interval $[0, 1]$; Others aim at estimating the *guess number* (e. g., [46, 59, 71]), i. e., are integer-valued; Most meters deployed at websites output a *textual description* of the password strength, e. g., [`Too short, Weak, Fair, Good, Strong`] for Google's PSM, in this case we convert these textual descriptions to natural numbers between 1 and the number of classes.

PSMs can be either *informative* when they are used merely to inform the user about the strength of the password (nudging the user towards more secure choices), or *enforcing* when passwords that are considered weak are not accepted by the system. Most deployed systems we analyzed, fall actually in the middle, enforcing a certain minimal strength, and informing (and nudging) the user towards more secure passwords beyond those minimal requirements.

3.2 Measuring Accuracy

Accuracy is one of the central factors of PSMs, and several PSMs have been proposed over the past few years. However, little work was done towards a fair comparison of different meters, and even on the question what constitutes a fair comparison, there is no agreement.

The preferred method to measure the accuracy of a strength meter is by comparing it to an ideal *reference*, measuring the *similarity* between the reference and the meter output. This idea is based on the intuition that weak passwords are those that are common and have been used before [13, 65, 71]. However, the techniques for comparing reference and tested meters in previous work were ad-hoc and ranged from measures counting overestimation errors to rank correlation metrics. In the following, we will systematically study which measures are most suited for performing this comparison. Specifically, we will show that previously used similarity measures have significant shortcomings limiting their validity and usefulness.

Before discussing specific similarity measures, it is instructive to consider properties that these measures should fulfill. To this goal, we specify which differences the meter and the reference should yield high and low similarity. There is no absolute truth in which requirements are desirable or not, and for specific applications, there may be additional requirements that are desired. We provide a list of requirements based on extensive experience with passwords and PSMs, and believe it captures requirements suitable for common online use.

By explicitly stating the desired requirements the selection process becomes much more transparent, and we will see that most previously used similarity measures fail even to fulfill some fundamental requirements, highlighting the importance of a systematic treatment. (Specific test cases derived from these abstract requirements are provided in the following section.)

(1) *Tolerance to Monotonic Transformations:* The output score given by strength meters is often not directly comparable. Their score can be based on the *number of guessing attempts*, different forms of *entropy*, on arbitrary scales like [`Weak, Fair, Strong`] vs. [`Terrible, Weak, Good, Excellent, Fantastic`], and other home-brewed measures of strength. Assuming that the underlying sorting of passwords is identical, these differences can be modeled as monotone functions. A good similarity measure should tolerate such monotone transformation and assign high similarity to such transformed strength estimations.

(2) *Tolerance to Quantization:* A particular case of monotone transformations is quantization, e. g., strength meters that divide the reported values into a small number of bins, often three to five. A good similarity measure should tolerate such quantization. Note, with a very low number of bins, e. g., 2 bins [`reject, accept`], the comparison becomes less meaningful, and scores will typically be low, even for otherwise reasonable measures. In the case of an *enforcing* PSM, the strength policy becomes particularly interesting. In this case, all passwords that are not accepted effectively end up in the lowest bin (commonly called "Too short," "Too easily guessed," or "Too weak"). The stricter the policy is set, the larger this lowest bin gets, reducing the overall precision. A good similarity measure should tolerate moderately large reject-bins.

(3) *Tolerance to Noise:* Small deviations in the strength estimations are frequent, based on slight differences in the used models, the training data, or other factors. A good measure should tolerate such minor deviations.

(4) *Sensitivity to Large Errors:* While small differences don't have a significant effect on the usefulness of a strength meter, large deviations, in particular, overestimates, can harm. A good measure needs to be sensitive to large variations in strength even for a small set of passwords.

(5) *Approximation Precision:* A similarity score is easier to compute and thus more useful if it doesn't need full knowledge of the meter. Specifically, strength meters deployed on websites put limits on the number of samples one can handle, either by the slowness of the process or more specific restrictions, like the number of allowed queries. Thus, a good measure should be easy to approximate from a limited number of samples.

Table 1: Evaluated Datasets

Name	Year	Service	Policy[1]	$\tilde{H}_\infty$	$\tilde{G}_{0.25}$
RockYou	2009	Social Games	5+	6.81	15.89
LinkedIn	2012	Social Network	6+	7.27	19.08
000Webhost	2016	Web Hosting	6+ [a-Z][0-9]	9.26	20.69

4 EVALUATED PASSWORD DATASETS

In this section, we discuss factors that influence password choice, introduce the password datasets that we will use to evaluate a broad selection of PSMs, and describe our *reference* password distribution which we will use for comparing different accuracy metrics.

4.1 Influencing Factors

When evaluating strength meters one must consider that password strength is contextual and influenced by many factors [4, 23, 50].

(1) Password leaks originating from a single web service follow a distribution partially specific for this site. For example, the password "linkedin" appears in the LinkedIn leak with a probability of 0.12 %, but does not appear in the RockYou leak. In contrast, the password "rockyou" appears with a probability of 0.06 % in the RockYou leak, but only appears with a probability of 0.000028 % in the LinkedIn leak. Often passwords from a service reflect the category of the service and include the name or semantic theme of the service [67].

(2) Website administrators often enforce password composition policies [42] (e. g., requiring the password to contain a digit or symbol or to be of a certain length) that force users into choosing different passwords which are compliant with the respective policy.

(3) Florêncio et al. showed that not using any weak passwords or not considering to reuse some passwords becomes impossible with a growing number of accounts. If no password manager is used, account grouping and reusing passwords becomes the only viable solution [25]. Given a fixed time-effort budget [5] it is sub-optimal to spend the same amount of effort for all accounts. Florêncio et al. [24] proposed to classify accounts into categories from "don't-care" to "ultra-sensitive" accounts based on, e. g., the consequences of account compromise.

(4) A password strength meter might be tuned and more intensively tested with a specific password leak. Specifically, academic meter proposals, which are based on probabilistic password models, require a lot of real-world password data. Some strength meters even include small blacklists of very common passwords.

While it is difficult to avoid all factors, we try to minimize their influence by testing three very different datasets in our experiments that differ by service, policy, and leak date. We selected the datasets to allow easy verification and generate reproducible results based on publicly available data. Our findings are limited to predominantly English speaking users and their password preference. To reason about the strength of a password distribution considering a best-case attacker, we provide the Min-entropy H_∞ as lower bound and partial guessing entropy (α-guesswork) G_α for alpha 0.25 as described by Bonneau [8].

[1]We list the active policy, at the time when the data breach happened.

4.2 Datasets

An overview of the used datasets that are described in the following is given in Table 1:

- RockYou: This is a well-established leak used extensively in previous work. 32 million plaintext passwords leaked from the RockYou web service in December 2009, via an SQL injection attack, which means that no bias was introduced. We include RockYou in our evaluation because of its popularity in the community. However, its passwords should be considered relatively *weak* ($\tilde{G}_{0.25} = 16$ bits).
- LinkedIn: The social networking website LinkedIn was hacked in June 2012. The full leak became public in late 2016. The leak contains a SQL database dump that includes approx. 163 million unsalted SHA-1 hashes. In the following, we use a 98.68 % recovered plaintext version resulting in approx. 161 million plaintext passwords. We expect the bias introduced by ignoring 1.32 % of (presumably strong) passwords to be low, as we are mostly interested in passwords whose probability can reasonably be approximated by their count. We include LinkedIn in our evaluation because we consider those passwords to be a reasonable candidate for *medium*-strong passwords ($\tilde{G}_{0.25} = 19$ bits).
- 000Webhost: Leaked from a free web space provider for PHP and MySQL applications. The data breach became public in October 2015. The leak contains 15 million plaintext passwords. Based on the official statement, a hacker breached the server, by exploiting a bug in an outdated PHP version, which again means that no bias was introduced. We include 000Webhost in our evaluation because of its enforcement of a lowercase and digits password composition policy, which results in a different password distribution containing relatively *strong* passwords ($\tilde{G}_{0.25} = 21$ bits).

To avoid processing errors in later steps (querying online meters), we cleaned the leaks, by removing all passwords that were longer than 256 characters or non-ASCII. This cleaning step removed 0.06 %, 0.09 %, and 0.19 % of the passwords from RockYou, LinkedIn, and 000Webhost.

4.3 Reference

To reason about various candidate metrics that might be suitable to measure the accuracy of a strength meter, we created a fourth dataset. The dataset only contains the frequent passwords of the LinkedIn leak. We have chosen LinkedIn because it was the largest available leak at our disposal. As has been shown by Bonneau [8] and Wang et al. [65], approximating strength for unlikely passwords is error-prone. To avoid such approximation errors, we limited the LinkedIn file only to include ASCII passwords that occur 10 or more times (count ≥ 10), which resulted in the reference password file containing approx. 1 million unique passwords.

We use the dataset as a) ideal *reference* and as b) strength meter output. For this, we divided the set into two disjoint sets REF-A and REF-B of about equal size by random sampling. In the following experiments, REF-A will be used as the reference, whereas REF-B will be used as a basis for the test cases, thus, simulates the meter output. The experiments as described in Section 5.2 operate on the count values; if the password `abc123` occurs 36, 482 times in LinkedIn, then REF-A and REF-B include a count value of ~18, 240 for this password. In Section 5.3 we report on the reliability of this

reference by performing additional tests that include uncommon passwords, as well as, the other leaks (RockYou and 000Webhost).

5 SIMILARITY MEASURES

In this section, we describe the process of selecting a suitable similarity metrics.

5.1 Test Cases

An overview of test cases described below is given in Table 2.

5.1.1 Monotonic Transformations. We prepared several cases to test a measures' tolerance to monotonic transformations: *DOUBLE:* For this test case we double the count values in REF-B. This represents the case that two strength meters use a different scale (e. g., one sets the cutoff for the `Strong` class at a different threshold than the other). This would naturally occur when two strength meters use the expected time to crack a password (such as zxcvbn [71]) but assume different speeds of the cracking hardware. *HALF:* For this test case we half the values in REF-B before applying the measure to calculate the similarity with the *ideal strength meter*. *LOG:* For this test case we take the logarithm to base 2 of the count values in REF-B. This occurs naturally when one strength meter reports strength in "expected number of guesses," and one in "bits of entropy." *SQR/SQRT:* Further, we added test cases by applying the square operation and the square root to REF-B, respectively.

5.1.2 Quantization. A substantial fraction of online meters uses binned output. Thus, such test cases are highly relevant in practice. *Q4-equi/Q10-equi:* For this test case we use quantization into four/ten bins, about the same number of passwords per bin (counting with multiplicities). *Q4-alt/Q10-alt:* Similar to the test case above, we use quantization into four/ten bins, but in this case, splitting into bins of equal size based on unique passwords (without counting multiplicities).

5.1.3 Disturbances. We have a number of test cases testing the tolerance and sensitivity to disturbances in the data. *RAND:* We use random values drawn from a uniform distribution between 1 and the maximum count value. This test case can be seen as a calibration of low similarities as any matching only happens by chance. *ADD-RAND:* We add small random disturbances to REF-B drawn according to a uniform distribution between 1 and the respective count of a password. *INV-WEAK-5:* We modify the weakest 5 % of passwords (with multiplicities), by setting their usually very large count to 0 (i. e., we invert their scoring to `very strong`). *INV-STRONG-5:* We modify the strongest 5 % of passwords (with multiplicities), by setting their usually very small count to the maximum count value (i. e., we invert their scoring to `very weak`).

5.2 Testing Different Metrics

Next, we describe a number of similarity measures and evaluate them on the test cases defined above to understand their properties and usefulness. The results are shown in Table 2, we will discuss these results in-depth in the remainder of this section.

5.2.1 Correlation. A straightforward way to measure similarity, which has been used in most prior work, is the correlation between the reference and the observed values.

Pearson Correlation Coefficient: The probably best known correlation measure. It is defined as the covariance divided by both standard deviations. Pearson correlation has several problems as a similarity measure for PSMs: First, it is sensitive to monotonic transformations (e. g., correlation of REF-A and LOG is 0.13), which is undesirable. Even worse, it is highly sensitive to quantization (which we typically encounter for most web-based meters), the correlation between REF-A and quantized versions Q4-equi/Q10-equi/Q4-alt/Q10-alt is close to zero (between approximately 0.1 and 0.05). Another issue is that it does not capture well the case INV-STRONG-5, where 5 % of strong passwords are given a weak score (arguably not a big problem at all), yet the similarity drops to around zero (−0.02). Two properties of Pearson correlation underlay this undesirable behavior. First, it is a parametric measure and computed from the given values (instead of, e. g., ranks such as Spearman correlation), which makes it sensitive to non-linear transformations of the data. Second, it gives each data point equal weight, even though the weak passwords have a much higher count (by definition), thus Pearson correlation weights deviations for strong passwords stronger, relatively speaking.

Spearman Rank Correlation Coefficient: It is defined as Pearson correlation over the ranks of the data. Thus it is based on ranks of the (sorted) data only. Spearman correlation has been used by previous work on password strength [13, 65]. Spearman is robust against monotonic transformations and quite tolerant to quantization, which is an improvement over Pearson. Still, it gives too much weight to strong passwords, similarly to Pearson correlation. One additional problem is visible for Spearman: the correlation between REF-A and REF-B should be (close to) 1, as we expect perfect correlation, however, Table 2 shows a correlation of 0.73. The underlying reason is again the missing weights, which leads to the situation that the strong passwords dominate the similarity score (around 50 % of passwords have a count of less than 20 in REF-A), and the (small) errors from sampling on those strong passwords pull the score from 1 (what would be expected) to around 0.7.

Kendall Rank Correlation Coefficient: Kendall's tau coefficient is quite similar to Spearman correlation, but conceptually simpler (it only takes into account if ranks are wrong and the direction, but not how big the difference is). Previous work, in fact, showed very similar results for Spearman and Kendall [65]. However, it has the disadvantage that naïve implementations (as in standard R) are computational expensive for larger samples requiring $O(N^2)$ operations. While Kendall is expected to be robust to monotonic transformations, a problem similar to Spearman reduces correlation to 0.56. Furthermore, adding randomness (ADD-RAND) introduces enough variation to reduce the similarity to 0.54, and the impact of quantization is stronger than for Spearman.

5.2.2 Weighted Correlation. One common problem with the above correlation measures is that they treat frequent and infrequent passwords as equally weighted data points, i. e., an error in a single infrequent password is rated equally as an error in a frequent password which influences much more accounts. Weighted correlation measures give specific weights to the data points, which we take to be the frequency in the reference dataset. (To the best of our knowledge, neither weighted Pearson nor weighted Spearman correlation has been used to compare PSMs before.)

Table 2: Comparing REF-A with *modified* REF-B using various similarity measures.

			(Weighted) Correlation Metrics					(Weighted) Mean Error Metrics						(Weighted) One-Sided/Pairwise Error Metrics							
	Test Cases	**Sim.**	**Pear.**	**Spear.**	**Kend.**	**wPear.**	**wSpear.**	**MAE**	**MSE**	**rMAE**	**rMSE**	**wrMAE**	**wrMSE**	**wrLAE**	**wrLSE**	**PE**	**PE-5**	**PU**	**wPE**	**wPE-5**	**wPU**
	REF-B	H	1.00	0.73	0.56	1.00	**0.99**	4	54	0.16	0.05	2.77	13	1.39	7	1.00	0.68	0.95	0.96	0.24	0.99
Monotonic Tra.	**DOUBLE**	H	1.00	0.73	0.56	1.00	**0.99**	28	300402	0.16	0.05	2.77	13	1.39	7	1.00	0.68	0.95	0.96	0.24	0.99
	HALF	H	1.00	0.73	0.56	1.00	**0.99**	14	75703	0.16	0.05	2.77	13	1.39	7	1.00	0.68	0.95	0.96	0.24	0.99
	LOG	H	0.13	0.73	0.56	0.49	**0.99**	24	301526	0.16	0.05	2.77	13	1.39	7	1.00	0.68	0.95	0.96	0.24	0.99
	SQR	H	0.93	0.73	0.56	0.99	**0.99**	3.E+05	7.E+16	0.16	0.05	2.77	13	1.39	7	1.00	0.68	0.95	0.96	0.24	0.99
	SQRT	H	0.45	0.73	0.56	0.96	**0.99**	23	300014	0.16	0.05	2.77	13	1.39	7	1.00	0.68	0.95	0.96	0.24	0.99
Quantization	**Q4-alt**	H	0.05	0.89	0.78	0.08	**0.73**	25	301711	0.10	0.02	5.41	18803	4.34	18799	1.00	0.69	0.75	1.00	0.81	0.75
	Q10-alt	H	0.06	0.91	0.80	0.09	**0.86**	22	301465	0.09	0.01	2.79	3006	1.86	3003	1.00	0.54	0.90	1.00	0.20	0.90
	Q4-equi	H	0.12	0.72	0.61	0.21	**0.97**	26	301766	0.16	0.04	3.76	29	2.58	25	1.00	0.78	0.36	1.00	0.36	0.75
	Q10-equi	H	0.11	0.90	0.80	0.25	**0.99**	25	301607	0.09	0.02	1.83	6	0.96	3	1.00	0.57	0.71	1.00	0.19	0.90
Disturbances	**RAND**	L	0.00	0.00	0.00	0.12	**0.04**	3.E+05	9.E+10	0.33	0.17	22.78	1.E+05	20.37	1.E+05	1.00	0.90	1.00	1.00	0.92	1.00
	ADD-RAND	H	0.99	0.70	0.54	1.00	**0.99**	15	29799	0.17	0.05	3.09	16	1.62	9	1.00	0.70	0.97	0.97	0.26	0.99
	INV-WEAK-5	M	0.25	0.73	0.56	-0.04	**0.70**	6	283087	0.16	0.05	5.52	1.E+06	4.13	1.E+06	1.00	0.68	0.95	1.00	0.29	0.99
	INV-STRO-5	M	-0.02	-0.13	0.01	0.50	**0.72**	1.E+05	7.E+10	0.37	0.19	14.93	28742	12.89	28727	1.00	0.96	0.92	1.00	0.99	0.98

Sim.: Expected similarity with REF-A; L=Low similarity, M=Medium similarity, H=High similarity.

Weighted Pearson Correlation: Is defined as (normal) Pearson correlation but weighting each data point with a weight vector, where we use the reference (REF-A) as the weight vector. This similarity measure exhibits similar problems as unweighted Pearson correlation for monotonic transformations, as expected, even though the effect is less pronounced, and remains highly sensitive to quantized data.

Weighted Spearman Correlation: Is defined as weighted Pearson correlation on the ranks. It is the most promising similarity measure considered so far. As ordinary (unweighted) Spearman correlation, it handles monotonic transformations well. It gives a correlation close to 1 to the test case REF-B (and the other monotone transformations including most quantized test cases, which was problematic before), as now the weights prevent the overrepresentation of strong passwords. It also handles the INV-WEAK-5 and INV-STRONG-5 cases well, where it assigns roughly the same correlation to both cases of around 0.7, a moderate but noticeable lower value than 1.

5.2.3 Mean Error. Another set of similarity measures is mean square error (MSE) and related concepts. We tested variations, inspired by the above results and techniques used in previous work.

Mean Absolute Error (MAE): Is defined as the average absolute error, with equal weight for each data point. A similar measure was used recently [71], where a logarithmic error was used. Our test cases reveal the following problems: It is highly sensitive to monotonic transformations, even linear ones (and previous work [71] needed to adapt the scales of the meters to get a reasonable comparison). Large deviations in the rating for single passwords only have moderate impact on the similarity (due to taking absolute errors only). Its sensitivity to deviations in frequent passwords is low (the error for INV-WEAK-5 is 6, only marginally larger than the error due to random sampling (REF-B with an error of 4).

Mean Squared Error (MSE): Is defined as the average over the squared error, giving more weight to large deviations. Properties of MSE are very similar to that of MAE.

Ranked Mean Absolute/Squared Error (rMAE/rMSE): Here we first rank the data (assigning ties the average rank), and compute the MAE or MSE of the ranks. As expected, the resulting measures are resistant to monotonic transformations. However, as they are non-weighted, they fail to capture errors for few frequent passwords (INV-WEAK-5). This means, in the bad performing PSM test case (INV-WEAK-5) both, rMAE and rMSE fail to show any difference to the reference making them unsuitable.

5.2.4 Weighted Mean Error. All error measures discussed in the previous subsection are unweighted, and thus fail to capture errors in few frequent passwords. In this subsection, we consider weighted variants.

Weighted and Ranked Mean Abs./Sq. Error (wrMAE/wrMSE): When we use both ranked and weighted data points, the resulting similarity measure becomes more discriminative, e. g., it allows to distinguish the INV-WEAK-5 and DOUBLE test cases. Both measures work very well on our test-cases (remember that lower values mean more similarity) and seem to be a reasonable choice.

5.2.5 One-Sided Errors. As described before, password strength approximations can be under- or overestimates. Previous work [46, 71] observed that a meter underestimating the security of strong passwords (e. g., INV-STRO-5) is less problematic than overestimating the strength of weak passwords (e. g., INV-WEAK-5). The former results in a user simply selecting another (presumably secure) password, whereas in the latter case the user believes having selected a secure password, where in reality it is weak.

Weighted and Ranked Mean Abs./Squared One-Sided Lower Error (wrLAE/wrLSE): One can define versions for MAE/MSE that only take one-sided errors into account. If this measure operates on count values, this approach favors meters that generally underestimate security: a meter that rates all passwords insecure (i. e., a high count value) will get a high rating. This can be prevented by operating on ranked data. On the tested datasets and test cases the resulting measures, wrLAE/wrLSE perform similarly to their two-sided versions wrMAE/wrMSE. A likely explanation is that wrLAE/wrLSE operate on ranked data. Therefore, overestimating the strength of one password generally leads to underestimating the strength of another password. (Weights and squaring differences (wrLSE) mean that the results still can differ, however, these effects seem to even out on the dataset that we considered.) For applications that call for one-sided metrics, one should consider non-ranked similarity metrics at the cost of losing the ability to tolerate monotonic transformations.

5.2.6 Pairwise Errors. In preliminary tests, we observed that several similarity measures give a low similarity score to quantized data. This behavior is undesirable, as heavy quantization loses information about the distribution.

We tried to address this problem by designing a similarity score that is based on two individual metrics: an *error metric* which describes how many passwords are not in the "correct" order, and a *utility metric* which describes if the meter provides "useful" and discriminative output. (To illustrate this problem, consider a strength meter with binary output, where only a few very strong passwords are "accepted," and the other passwords are "rejected". This meter would have a low error rating, as it mostly preserves the order, but a low utility rating, as most passwords are in the same bin). This mechanism is based on the rank. We evaluated several variants of this basic idea.

Pairwise Error/Utility Rate (PE/PU): These consider the relative ranking of all pairs of passwords. PE considers the fraction of pairs where the meter and the reference disagree (where a tie in one of the two is not counted as a disagreement), whereas the PU considers the fraction of pairs where the meter sees a tie. (A meter outputting the same strength for all passwords, i. e., uses a single bin, has a PE of 0, but also a PU of 0.)

Pairwise Error Rate More Than 5 % (PE-5): As small deviations are typically considered a non-problem, for this variant we tolerate any deviation that is less than 5 % (in terms of rank) and do not count them towards the error.

5.2.7 Weighted Pairwise Errors. We have argued before that unweighted measures not taking into account the specific probabilities of passwords systematically bias results.

Weighted Pairwise Error/Utility Rate (wPE/wPU)/Weighted Pairwise Error Rate More Than 5 % (wPE-5): We use weighted versions of the three measures introduced before, where we weight each pair with the product of the probabilities of the two passwords.

Implementation: All measures are implemented using R v3.4.4 (March 2018). For Pearson and Spearman, we use standard R. For weighted Pearson and Spearman we use the *wCorr* package[2]. For calculating the Kendall correlation, we use a $O(n \log n)$ optimized version from *pcaPP*[3].

5.3 Reference Validation

To confirm our findings and test the reliability of our reference, which is based on the common LinkedIn passwords, we repeated our analysis using RockYou and 000Webhost. The leaks are different in size; thus the resulting number of passwords tested were different. While the reference had approx. 1 million passwords that occurred 10 or more times, RockYou only includes 250 000 and 000Webhost 62 000 unique passwords. Across different leaks we observed only minor differences. The tendencies for correlation, mean error, one-sided error, and pairwise error metrics, which can be observed in Table 2, remain the same independent of the tested password leak. For example, for the three leaks the wSpear. metric results vary only around ±0.04 across all test cases.

Furthermore, we repeated our tests with a LinkedIn set that included *uncommon* passwords (count ≥ 2). Including uncommon passwords is expected to be more error prone [8, 65]. While the common variant included approx. 1 million passwords, the uncommon version consisted of 31 million unique passwords. However, our results show that the tendencies from Table 2 remain the same. For example, for the uncommon variant the wSpear. metric results vary around ±0.07 across all test cases.

To summarize, in those additional tests we found only minor differences in the behavior of the similarity measures across password leaks. Moreover, including uncommon passwords had a bigger albeit overall negligible impact on the results.

5.4 Recommendation

We report results for 19 candidates for similarity measure. We considered 5 correlation-based similarity measures, 6-variants that are mean absolute/square error-based, as well as, 8 one-sided/pairwise error metrics and evaluated them on a number of test cases. Those tests included commonly observed cases like logarithmic transformation and quantization, but also meters that incorrectly judge strength simulated via disturbances.

We have seen that measures that are not weighted largely fail to capture essential aspects of the (highly skewed) distributions of passwords. Consequently, sensible measures should be weighted. Furthermore, we observed that measures based not on rank (but rather on values) are generally too sensitive to monotonic transformations and quantization to be useful.

In our evaluation the metrics wSpear., wrMAE, wrMSE, wrLAE, wrLSE, and wPE-5/wPU are weighted and ranked metrics that performed well and seem suitable as comparison metric. For the remainder of this work we have selected weighted Spearman correlation. It is not perfect, especially on quantized output, and it does not differentiate between under- and overestimating strength, but performed well on most test cases. Furthermore, it is a standard metric and easy to interpret, relatively good to approximate from sampled data (cf. Section 5.5), and implementations are easily available. Also, (unweighted) Spearman correlation has been used before to evaluate strength meters [13, 65].

5.5 Sampling

Collecting data from online sources is often cumbersome (e. g., previous work [16] that evaluated data from (online) password strength meter went through great effort to collect large amounts of data). Therefore, we want to determine confidence intervals for our measures to select the amount of data we need to collect. Determining accurate bounds is non-trivial, and to the best of our knowledge, no bounds are known that are applicable to our problem.

We determine empirical confidence intervals for the weighted Spearman measure (as it was the most promising one in the previous section) by repeated sub-sampling from the reference REF-A and the test cases. We sample subsets of varying sizes, ranging from 100 to 10,000, and computing similarity on those subsets, using the full data available to determine the strength score of the reference.

[2]Package: wCorr (Weighted Correlations), Version 1.9.1, May 2017, https://cran.r-project.org/package=wCorr, as of September 10, 2018
[3]Package: pcaPP (Robust PCA by PP), Version 1.9-73, January 2018, https://cran.r-project.org/package=pcaPP, as of September 10, 2018

Table 3: (Empirical) confidence intervals for REF-A vs. Q4-equi/LOG for different sample sizes and the weighted Spearman similarity measure. All are determined using 10,000 iterations, and a 5 % confidence level. Given is the width of the confidence interval, as well as the boundaries (in brackets).

# Samples	100	500	1000	5,000	10,000
Q4-equi	0.146 [0.852 , 0.998]	0.069 [0.916 , 0.985]	0.044 [0.928 , 0.972]	0.022 [0.944 , 0.966]	0.027 [0.940 , 0.966]
LOG	0.081 [0.919 , 1.000]	0.033 [0.966 , 0.999]	0.024 [0.974 , 0.998]	0.013 [0.983 , 0.996]	0.011 [0.985 , 0.997]

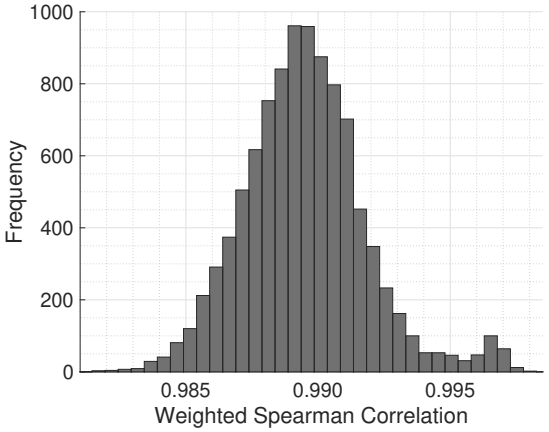


Figure 1: Histogram example for the monotonic transformation error LOG using weighted Spearman correlation, 10,000 samples, and 10,000 iterations.

We repeat this process 10,000 times and determine the interval that contains 95 % of all similarity values (with 2.5 % larger and 2.5 % smaller). We report both the width of the interval and the actual interval. We perform this process for two different datasets, namely for Q4-equi and for LOG. While this does not give a formal guarantee that the actual value can be found in this interval, it gives us reasonable confidence and determines rough boundaries. Note that this process only takes into account (random) errors caused by sampling; it does not take into account any systematic errors that may be introduced, e. g., by the smaller sample size. The summary of results is shown in Table 3.

An example of a histogram of the correlation values is given in Figure 1, which was determined for weighted Spearman correlation with 10,000 samples and 10,000 iterations. The histogram has a median of 0.990, min./max. of 0.980/1.000, and 2.5 %/97.5 %-percentiles of 0.985/0.997, resulting in a width of the 95 % confidence interval of 0.011. We see that, as expected, the width of the confidence interval decreases with increasing sample size. For weighted Spearman, we find widths of 0.027 and 0.011, respectively, and we will assume differences greater than 0.05 to be significant.

6 EVALUATION

Guided by practical requirements, we distinguish between two different application scenarios.

(1) PSMs deployed to protect online accounts, i. e., the most prevalent online logins for social networks, email providers, etc. For online accounts, the operator can and should implement measures to limit the effectiveness of online guessing attacks such as rate-limiting. Typically one considers between 100 and 1000 allowed guesses within 30 days [9, 28, 31, 65, 66].

(2) Strength meters deployed to protect local authentication, such as hard disk encryption. In this scenario the number of guesses the adversary can test is only limited by the computational power of the adversary; In real-world attacks, the number of guesses per day on a single GPU is in the order of 10^9 to 10^{12} guesses [30]; some even consider up to 10^{14} guesses to be reasonable [26].

6.1 Online Guessing

For online account PSMs, techniques such as rate-limiting can reduce the risk of online guessing attacks. Based on previous work [65, 66], which describes 1000 guesses as a reasonable limit an attacker can perform in an online guessing attack, and based on the assumption that the attacker is acting rational and guesses the most likely passwords first, we deduce that the most interesting set of passwords relevant for this kind of attack is the most likely 10,000 passwords. If each user omits passwords from the "easier half" of this set, then overall security will greatly be improved.

Sampling strength meter scores for these 10,000 passwords of all three datasets (RockYou, LinkedIn, 000Webhost) would put an unnecessary burden on the server infrastructure, and might even trigger server-side alerts. Given the results on sampling accuracy in Section 5.5 we try to avoid such implications by restricting ourselves to 1000 samples from these online services: i. e., out of the 10,000 most common RockYou passwords, we uniformly random sample 1000 passwords. We repeat this process for LinkedIn and 000Webhost, respectively, to obtain three different *online* guessing datasets. For the most likely passwords, we have very accurate frequency estimates. So for those common passwords, we can use the sample frequency as a ground truth for their strength in an online attack.

6.2 Offline Guessing

For offline guessing attacks, there is no limit on the number of attempts an attacker can perform, depending on the computing capabilities and the password hashing function deployed for storing the password. Consequently, the sample frequency in the datasets does not provide useful information about the strength in an offline attack, as the number of guesses (by far) exceeds the size of the dataset. Instead, we use the performance of common guessing tools as the reference. More specifically, we use the results of the Password Guessability Service (PGS) [62], which allows researchers to send lists of passwords (in plaintext), and the service evaluates when these passwords will be guessed by common configurations of widely used password guessing tools. Work by Ur et al. [62] found that the attribute *min_auto* is a good measure

of resistance to guessing attacks, even with humans password experts involved. We use this recommended configuration without a password composition policy (1class1) as the ground truth for the offline guessing evaluation. For each of the three datasets (RockYou, LinkedIn, 000Webhost) we sampled 10,000 passwords to obtain the three different *offline* guessing datasets.

6.3 Selected Meters

Academic Proposals. We considered different meter proposals from the literature.

- *Heuristic/NIST*: In 2004 the NIST published SP 800-63 Ver. 1.0 [11], which includes heuristics based on length and compliance to a composition policy on entropy estimation. The heuristic also considers a bonus if the password passes a common dictionary check. The latest version, SP 800-63B [31] from June 2017, no longer includes the ad-hoc heuristic.
- *Markov Model/OMEN*: In 2012 Castelluccia et al. [13] proposed to train n-gram Markov models on the passwords of a service to provided accurate strength estimations. The estimation is thus based on the probabilities of the n-grams a password is composed of. The meter provides adaptive estimations based on a target distribution but is limited to server-side implementations.
- *Heuristic/Comp8*: In 2012 Ur et al. [61] investigated how PSMs can be used to nudge users towards stronger passwords. They outlined a scoring algorithm derived from a composition policy called "Comprehensive 8." Due to the lack of better alternatives, this scoring function was used to estimate the strength of a password. While this LUDS-based approach should no longer be used, we include it for completeness.
- *Heuristics/zxcvbn*: In 2012 Daniel Wheeler proposed a PSM in a Dropbox Inc. blog post. It is based on advanced heuristics that extend the LUDS approach by including dictionaries, considering leetspeak transformations, keyboard walks, and more. Due to its easy to integrate design, it is deployed on many websites. The meter was recently backed up by scientific analysis [71].
- *PCFG/fuzzyPSM*: In 2012 Houshmand and Aggarwal [34] proposed a system to analyze the strength of a password. For this, they used a PCFG-based approach. In 2016 Wang et al. [65] extended the concept by proposing a *fuzzy* PCFG to model password strength, based on which mangling rules are required to modify a basic dictionary to match a training distribution of stronger passwords.
- *Heuristic/Eleven*: In 2013 Egelman et al. [22] studied password selection in the presence of PSMs. For their meter, they decided to use a similar metric for strength as NIST. Similar to LUDS approaches the meter considers character set sizes and length.
- *RNN/*DD-*PSM*: In 2016 Melicher et al. [46] proposed to use a recurrent neural network for probabilistic password modeling. For our analysis, we use the guess number estimations provided by the RNN. The authors also describe a method that allows a client-side implementation using a special encoding and a Bloom filter. In 2017, Ur et al. [59] extended the concept by adding *data-driven* feedback using 21 heuristics that try to explain how to improve the password choice. We use Ur's website[4] for additional measurements.
- *Heuristic/LPSE*: In 2018 Guo et al. [32] proposed a lightweight client-side meter. It is based on cosine-length and password-edit distance similarity. It transforms a password into a LUDS vector and compares it to a standardized *strong-password vector* using the aforementioned similarity measures.

Password Managers. We also tested meters that protect high value encrypted password vaults. If no further protection mechanism is deployed [27], especially the security of cloud-based password managers depend on the use of a high entropy secret. Thus, service providers need to give accurate strength estimates. Furthermore, vaults that offer the ability to store user-chosen credentials might show strength estimates for their stored secrets, too. For our work, we analyzed the PSMs of 11 popular password managers, including *1Password* [2], *Bitwarden* [1], *Dashlane* [14], *Enpass* [55], *KeePass* [51], *Keeper* [39], *LastPass* [43], and more.

Popular Websites. We queried password strength meters from popular web services within the top 100 ranking published by Alexa Internet. Our samples include large sites like *Apple*, *Baidu*, *Dropbox*, *Facebook*, *Google*, *Microsoft*, *reddit*, *Twitter*, *Sina Weibo*, *Yandex*, and more. For better comparability, we tried to include sites that were queried in previous work by de Carné de Carnavalet and Mannan [16]. The authors published their findings on all the meter codes via a "Password Multi-Checker Tool" on a self-hosted website [17], allowing one to compare their results with the currently implemented versions.

Operating Systems. We analyzed password strength meters from standard operating systems. Microsoft's Windows and Apple's macOS do not provide a strength estimation during account creation. However, Apple includes a *Password Assistant* with a strength estimation functionally that is used by the *Keychain Access* application while generating passwords, e. g., for file encryption. Canonical's Ubuntu distribution shows a PSM during the account creation and hard disk encryption setup. It is part of the graphical live CD installer *Ubiquity* and based on Mozilla's Seamonkey PSM function.

6.4 Querying Meters

To query the website PSMs, we used similar techniques as previous work [16]. For JavaScript and server-side implementations, we used the *Selenium* framework [35] to automate a headless Google Chrome browser. As all JavaScript (that involves no server communication) is evaluated on the client only, one can obtain large quantities of strength estimates in a short period. The used browser automation approach allows to execute JavaScript, thus intermediate results that are not displayed in the GUI of the meter, but exist in the Document Object Model (DOM), are accessible, too.

For the academic proposals, a more evolved approach was required, as some meters require a training or no implementation was available. For the training, we sampled 10 million passwords from each sanitized dataset (excluding the respective *offline* and *online* passwords). Please note, not all meters make full use of all available training data, e. g., fuzzyPSM, OMEN, and the RNN-based approach have specific requirements. For example, the Markov model approach used by OMEN does not allow training passwords shorter than the *n*-gram size. Similarly, fuzzyPSM does not require a training corpus larger than 1 million passwords. For the RNN-based approach we were forced to limit the training set to passwords no longer than 30 characters.

[4] Data-Driven PSM: https://cups.cs.cmu.edu/meter/, as of September 10, 2018

Table 4: We computed the weighted Spearman correlation as the best similarity score (cf. Section 5). The table lists the online use case on the left, the offline use case on the right. We highlighted if and on how many bins a meter quantized its output and list whether a meter runs on client- or server-side.

				Online Attacker			Offline Attacker		
ID	**Meter**	**Type**	**Quant.**	**RockYou**	**LinkedIn**	**000Webhost**	**RockYou**	**LinkedIn**	**000Webhost**
1A	**Comprehensive8** [61]	C	-	-0.652	-0.589	0.251	-0.476	-0.616	0.441
2	**Eleven** [22]	C	-	0.670	0.912	0.492	0.755	0.951	0.733
3	**LPSE** [32]	C	Q3	0.584	0.669	0.508	0.544	0.718	0.693
4C	**Markov (Multi)** [27]	S	-	0.721	0.998	0.902	0.997	0.995	0.777
5B	**NIST (w. Dict.)** [11]	C	-	0.669	0.910	0.472	0.756	0.953	0.816
6	**PCFG (fuzzyPSM)** [65]	S	-	1.000	0.994	0.963	0.998	0.999	0.899
7C	**RNN Target** [46]	C	-	0.951	0.913	0.965	0.896	0.860	0.885
8A	**zxcvbn (Guess Number)** [71]	C	-	0.989	0.990	0.554	0.989	0.999	0.868

Type: C=Client, S=Server; **Quantization:** Q3–Q6=Number of bins, e. g., Q5=[`Terrible, Weak, Good, Excellent, Fantastic`];

For Comp8, LPSE, and fuzzyPSM we contacted the authors that kindly shared their source code or evaluated their implementation and shared the results with us. For the RNN PSM, we used an implementation by Melicher et al. [45]. We tested multiple variants: i) Based on the guess numbers of a pre-trained (generic) password distribution. ii) Based on a client-side JavaScript implementation (using a different password composition policy) [58]. iii) Based on the guess numbers of a self-trained RNN using a matching distribution (targeted), following the recommend construction guidelines. iv) Based on a self-trained RNN using a matching distribution (targeted) including a Bloom filter made of the top 2 million training set passwords (following the recommendations in the original paper [46]). For the Markov approach, we modified a version of the Ordered Markov ENumerator (OMEN) [3] by Dürmuth et al. [21] (a password guesser implementing the approach of Castelluccia et al. [13]) and used the aforementioned training set to obtain strength estimates. As this approach uses quantized (level-based) strength estimates only, we also implemented an approach described by Golla et al. [27] that outputs probabilities instead of quantized scores and increases precision by training one model per password length. For zxcvbn, we used the official JavaScript implementation [20]. For NIST we used a JavaScript implementation of the meter [15]. We built a dictionary consisting of the top 100,000 passwords from Mark Burnett's 10 million password list [10], which has been used as blacklist by previous work [33].

For most of the password managers (1Password, Bitwarden, Keeper, etc.), we were able to query their respective web interface version using Selenium. For RoboForm and True Key we automated the respective Chrome extensions using Selenium. For KeePass, we used the *KPScript* plugin on Windows [52]. For Dashlane we used the *Appium* framework [38] and its Windows Driver to automate the Windows desktop. While analyzing Enpass's PSM we found the use of the official zxcvbn implementation [54] (including the same dictionaries), thus we didn't query Enpass. Instead, we report the zxcvbn results.

For the operating systems, we queried Ubuntu's PSM using the original Python script from the *Ubiquity* source code [12]. For macOS we used *PyObjC* [49], a Python Objective-C bridge to query Apple's *Security Foundation* framework.

7 RESULTS

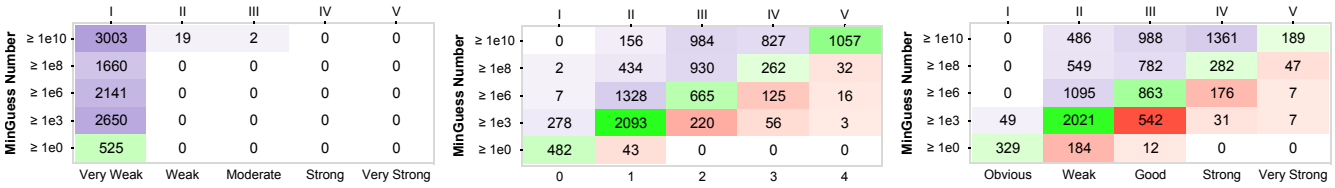
Next, we present and discuss the results of the evaluation of the different meters, both for the online and offline use case. Some of the academic PSM results are summarized in Table 4. We present the results for the online use case on the left, and for the offline use case on the right. We report results for both use cases for all strength meters, even though some are designed for one specific use case only (e. g., password meters deployed on websites are intended for the online use case). We computed the weighted Spearman correlation as the best similarity score selected in Section 5.

The full table for all 81 password strength meter variations can be found in the Appendix A. The primary results are separated into four categories (academic proposals, password managers, operating systems, and websites). A fifth category is included for comparison and is based on the "Password Multi-Checker Tool" [17] by previous work [16]. A version of our results that allows an easier comparison, provides bar charts of the quantizations, and more can be found online [29]. Please note, not all tested mechanisms like ID: 5A/B NIST or ID: 33 Have I Been Pwned? are intended to be used as a strength meter. Thus, the reported results for those estimators cannot be directly compared with others as their parameters can likely be augmented to perform better.

7.1 Overall Performance

The three best-performing *academic* meters are ID: 6 fuzzyPSM (0.899 – 1.000), ID: 7C RNN Target (0.860 – 0.965), and ID: 4C Markov (Multi) (0.721 – 0.998) for both online and offline use cases. A number of other PSM variants perform well, including ID: 8A zxcvbn (Guess Number) (0.554 – 0.999).

For *password managers* we found ID: 13A KeePass and ID: 14B Keeper to be accurately ranking meters (0.284 – 0.884). Furthermore, we found the zxcvbn-based ID: 17B RoboForm to be precise (0.528 – 0.962). Across the binning PSMs we found some of the zxcvbn (Score)-based meters, e. g., ID: 17A RoboForm (Q4), ID: 17C RoboForm Business (Q6), ID: 18 True Key (Q5), and ID: 12 Enpass (Q5) to be accurately ranking (0.341 – 0.827). The PSM in ID: 10A Bitwarden shows significant problems. Similar, but less pronounced are the inaccuracies of ID: 16B LogMeOnce and ID: 19A Zoho Vault. All three are LUDS-based meters.



(a) **ID**: 13B (KeePass), **wSpear**: 0.002 (b) **ID**: 8B (zxcvbn, Score), **wSpear**: 0.567 (c) **ID**: 41B (Twitter), **wSpear**: 0.665

Figure 2: Number of passwords per bin: An accurate and correctly binning PSM produces a diagonal green line. A meter that only assigns the weak/strong bin is visualized via a vertical bar on the left (purple)/right (red). The quantization degrades the precision, if bin thresholds are incorrectly chosen (cf. Figure 2(a)), the relative ranking is lost and the correlation degrades.

When it comes to PSMs used by current *operating systems*, we found very negative results. While macOS does not prominently display their PSM, Ubuntu uses the PSM during account creation and hard disk encryption. We found both meters to perform poorly. An analysis of Ubuntu's PSM source code revealed a LUDS meter. Their approach counts the number of uppercase, digit, and symbol characters and multiplies them with some magic constants. The estimation function is a re-implementation of Mozilla's *Seamonkey* meter which dates back to 2006. First bug reports about the poor quality and inconsistency in the assessment date back to 2012 [6].

While the weighted Spearman correlation decreases due to the effects of quantization (cf. Section 7.2), we observed a relatively good accuracy for some of the *website* PSMs, too. For example the non-binning ID: 35B Microsoft (v3), ID: 25A Best Buy, ID: 28A Drupal, and ID: 41A Twitter PSMs (0.424 – 0.951). Across the binning PSMs we found some of the zxcvbn (Score)-based meters, e. g., ID: 36 reddit (Q5) and ID: 40A Twitch (Q5) (0.197 – 0.817) and some non-zxcvbn-based PSMs like ID: 41B Twitter (Q5), ID: 32A Google (Q5), and ID: 35A Microsoft (v3) (Q4) to be accurately ranking (0.487 – 0.763). Based on our measurements ID: 33 Have I Been Pwned? performs excellently. This is likely owed to the fact that all tested datasets are part of the *Pwned Passwords list* [36]. Thus, different results are expected if non-breached passwords are evaluated. Surprising are the results of ID: 27A Dropbox (0.056 – 0.611), the developers of the *zxcvbn* meter. On their website, they rely on a Q4 score-based implementation (they discard the first bin, i. e., all passwords with a guess number below 10^3). Based on our ID: 8B zxcvbn (Score) findings, we expected better results. Note that the results of ID: 27B Dropbox using an older implementation by previous work [16] results in similar low performance.

To *summarize*, the academic contribution to strength estimation is outperforming many other meters and has brought up several concepts that are improving the estimations. Some other factors may contribute to those meters performing well: Specifically for the academic proposals we often have continuous scores, and the meters are trained on the distribution.

The PSMs in password managers perform reasonably well, with a few exceptions (ID: 10A Bitwarden, ID: 16B LogMeOnce, and ID: 19A Zoho Vault). Similar to previous work, we measured a good accuracy for ID: 13A KeePass. The high accuracy of ID: 17B RoboForm and others are explained by the use of zxcvbn. PSMs in operating systems are not popular, even though, when present, they are used for security-sensitive operations. Current implementations are LUDS-based meters that lack any helpful guidance and should be replaced. Website-based PSMs are doing reasonably well, with correlations up to 0.951. Most of our evaluated website PSMs are client-side JavaScript meters, also popular are hybrids. Server-side implementations were rare in our evaluation set. We speculate there are several reasons that websites are not using better meters: lack of awareness, lack of guidance on the quality of meters, and the usually larger size of academic meters that need to store the model parameters.

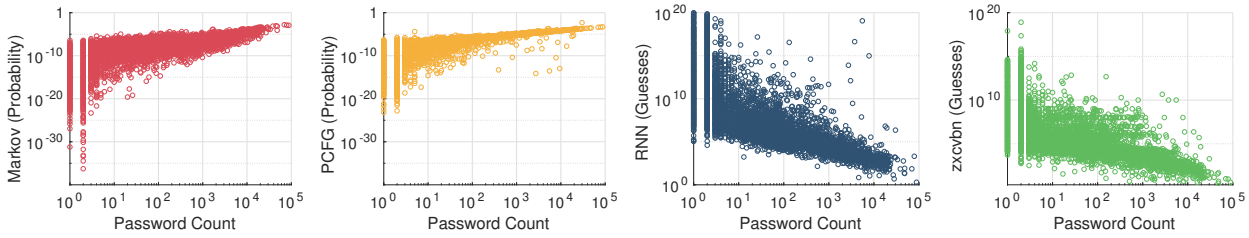
7.2 Effect of Quantization

Almost all PSMs on *websites* provide a binned output, as users rely on tangible feedback like `Weak` or `Strong` instead of a more abstract probability or guess number. Binning will reduce the accuracy of a meter. However, weighted Spearman is relatively robust against this effect. In the following, we qualitatively analyze the estimates of binning meters to provide a more intuitive way to compare weighted Spearman correlation with the over- and underestimates of the meters. For this, we use the PGS [62] *min_auto* guess number, and a logarithmic binning similar to zxcvbn (Score). The results are visualized in Figure 2.

7.2.1 The ≥ 10^3 / V Bin. This bin includes passwords that are weak ($10^3 \leq$ guess number $< 10^6$) but misjudged by the meter to be strong. An analysis of ID: 41B Twitter's bin revealed weakness in detecting keyboard walks and leet transformations. The password `!QAZxsw2` (a keyboard walk on US keyboards) as well as `P@$$w0rd`, `jessica#1`, and `password@123` were incorrectly ranked. ID: 8B zxcvbn (Score)'s bin includes `misdosamores` and `mardelplata` (film and city names), as well as `oportunidades` (common Spanish term).

7.2.2 The ≥ 10^{10} / II Bin. This bin includes passwords that are strong but misjudged by the meter to be somewhat weak. ID: 41B Twitter's bin revealed problems with digit-only passwords like `9371161366` that are usually cracked using a *Mask* attack. Furthermore, it includes all lowercase phrases like `itsababydog` that attackers crack by running a *Combinator* attack using two dictionaries. ID: 8B zxcvbn (Score)'s bin includes `zxcvbvcxz` (a variation of the meter's name giving keyboard walk) usually cracked via a dictionary or Mask attack. Additionally, we found phrases like `atlantasports`, which is likely cracked with a Combinator attack.

To further study the binning effect for real-world data, we look at those meters that provide both, quantized and non-quantized feedback. Interesting is the case of ID: 13B KeePass (Q5). While the



(a) Markov (b) PCFG (c) RNN (d) zxcvbn

Figure 3: PSM scatter plots: Increasing password counts on the x-axis. `Strong` (less common) passwords are on the left, `Weak` (more common) passwords are on the right. Estimated strength values (measured as probability/guess number) on the y-axis.

strength estimates are relatively precise ID: 13A KeePass (0.393 – 0.884), the binning had severe consequences. The PSM enforces very strong passwords (i. e., "Very weak" for score < 64 bit) resulting in the majority of passwords falling in the weakest bin. (*KeePass* does not display this textual feedback in their password manager software). In comparison, we see that for ID: 9A 1Password (0.276 – 0.807) the binning of the strength estimates by ID: 9B 1Password (Q5) (0.276 – 0.813) had close to no effect on the accuracy. There are cases where binning improves the score, e. g., the unbinned version ID: 10A Bitwarden (−0.635 – 0.676) performs substantially worse than the binned version ID: 10B Bitwarden (Q3) (0.258 – 0.725). The reason is that binning can eliminate some types of errors of a meter, depending on the precise binning boundaries.

7.3 Performance Over Time

It is interesting and illustrative to compare our results with those from de Carné de Carnavalet and Mannan [16], which were collected in June/July 2013. By analyzing the matching set of websites, one can observe positive and negative developments in the past 5 years. First of all, ID: 26A vs. 26B (China Railway) as well as ID: 31B vs. 31C (Fedex) did not change at all. Second, some meters most notably ID: 23A vs. 23C (Apple), as well as ID: 27A vs. 27B (Dropbox), and ID: 28B vs. 28C (Drupal) show a degraded ranking accuracy. When analyzing the Apple hybrid PSM, we found server-side blacklisting of the entire 14 million RockYou passwords in combination with a basic LUDS approach that checks for symbols and length. Finally, we can report a positive development for ID: 32A vs. 32B (Google), ID: 42B vs. 42C (Yandex), and ID: 41B vs. 41C (Twitter). Note that Twitter changed its quantization over time; thus results are not directly comparable. If we consider all websites in our evaluation set, one observes slightly better performing PSMs than in the complete set of 2013, but no significant change. One can observe that the majority of websites did change their PSM over time. (During our crawler development and data collection we observed how *reddit* changed its meter from a simple LUDS approach to zxcvbn.)

7.4 Recent Proposals and Future Directions

Recent academic proposals are among the best-scoring approaches. Specifically, instances of Markov models, PCFGs, RNNs, as well as zxcvbn, score excellently. Figure 3 shows the distribution of the strength estimations for the LinkedIn *offline* dataset. On the x-axis, one can see increasing password count numbers. This way, `strong` (less common) passwords are on the left; `weak` (more common) passwords are on the right.

While none of the approaches outperforms the others in terms of accuracy, other factors become more critical. For example to decrease the dependency of accuracy on the password distribution. We chose the different password evaluation sets to show the varying performance based on the (trained) distribution. A representative example for this is ID: 7B RNN Generic (Web) (0.421 – 0.777) vs. ID: 7C RNN Target (0.860 – 0.965). It shows the generic meter performance (using a different composition policy) in comparison to a targeted (distribution matching) trained variant.

Also relevant are the storage requirements of the meters: While the inaccurate and straightforward LUDS meters can fit into a couple of bytes, the (uncompressed) n-gram databases of meters based on Markov models can occupy hundreds of megabytes to gigabytes of disk space. The current database underlying the Have I Been Pwned? meter requires 30 GB (or enough trust to send partial password hashes to a third-party service). Optimized variants fit into 860 MB of memory using a Bloom filter [57]. zxcvbn has a size of around 800 KB. Decreasing the size of meters while maintaining high accuracy seems a worthwhile research goal. The RNN approach by Melicher et al. [46] is a first good example that reasonable accurate meters can fit into a couple of megabytes [46].

Furthermore, we need to better understand and mitigate the negative effects of quantization. While the non-binned academic proposals perform well, we need to find a way to transfer their accuracy to quantized output that is required for users. In the set of evaluated password strength meters we found score-based (i. e., > 42), percentage-based (i. e., > 75 %), and logarithmic (i. e., $\geq 10^6$) binning approaches, equal sized and unequal sized bins, magic constants, and rule-based binning.

Finally, what we have learned from the success of zxcvbn (which is implemented at several sites according to our findings) is that providing implementations in multiple programming languages that are readily deployable helps adoption.

7.5 Limitations

Albeit we carefully selected the datasets for our evaluation, we only simulated real-world password choice using breached passwords. As mentioned in Section 4 password distributions are influenced by many factors, thus the three evaluated sets only reflect a small set

of passwords. In particular, they reflect mostly an English speaking community. The impact of password-composition policies was not studied, primarily, because constraining the passwords in each leak to those that satisfy a given policy does not reflect real user behavior [40].

While weighted Spearman correlation was the best in our set of tested metrics for quantized strength estimations, it still is not perfectly accurate. Thus, results for quantized meters need to be interpreted carefully. In general, the lower the number of bins, the less precise the results, as explained in Section 5. Furthermore, different application contexts may place different demands on meters, thus may require different choices of similarity metrics.

Finally, measuring the accuracy alone is not enough to asses the overall performance of a meter. Usability and deployability aspects are vital for a complete assessment but are not presented in our analysis.

8 CONCLUSIONS

In this work, we considered the accuracy of password strength meters. We have demonstrated that the currently used measures to determine the accuracy of strength meters (such as Pearson and Kendall correlation) are not precise. We conducted a large comparison of different similarity measures and argued that weighted Spearman correlation is best suited to precisely and robustly estimate the accuracy of strength meters.

We applied this measure to 45 different strength meters and determined their accuracy for an online and offline use case. We found that the academic PSM proposals based on Markov models, PCFGs, and RNNs perform best. We also found several websites and password managers to have quite accurate strength meters. However, the strength meters used in practice are less accurate than academic proposals, and we see no significant improvement of meter accuracy when comparing with meters from 5 years ago.

High accuracy is one important aspect that impacts the security of a password strength meter. Also vital are usability and deployability aspects, those are independent of the presented work. We hope our work aids further improvements of PSMs and provides helpful guidance and a metric for the selection of accurate PSMs and thus helps to improve the situation of password-based user authentication.

REFERENCES

[1] 8bit Solutions, LLC. 2018. bitwarden (Web) – Free Open Source Password Manager. https://bitwarden.com, as of September 10, 2018.

[2] AgileBits, Inc. 2018. 1Password (Web) – Password Manager. https://1password.com, as of September 10, 2018.

[3] Fabian Angelstorf and Franziska Juckel. 2017. OMEN v0.3.0 - C Implementation of a Markov Model-based Password Guesser. https://github.com/RUB-SysSec/OMEN, as of September 10, 2018.

[4] Daniel V. Bailey, Markus Dürmuth, and Christof Paar. 2014. Statistics on Password Re-use and Adaptive Strength for Financial Accounts. In *Security and Cryptography for Networks (SCN '14)*. Springer, Amalfi, Italy, 218–235.

[5] Adam Beautement, M. Angela Sasse, and Mike Wonham. 2008. The Compliance Budget: Managing Security Behaviour in Organisations. In *New Security Paradigms Workshop (NSPW '08)*. ACM, Lake Tahoe, California, USA, 47–58.

[6] Sebastian Benvenuti. 2012. Ubiquity – Ubuntu Should Encourage Stronger Passwords. https://bugs.launchpad.net/ubuntu/+source/ubiquity/+bug/1044868, as of September 10, 2018.

[7] Matt Bishop and Daniel V. Klein. 1995. Improving System Security via Proactive Password Checking. *Computers & Security* 14, 3 (1995), 233–249.

[8] Joseph Bonneau. 2012. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In *IEEE Symposium on Security and Privacy (SP '12)*. IEEE Computer Society, San Francisco, California, USA, 538–552.

[9] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. 2012. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *IEEE Symposium on Security and Privacy (SP '12)*. IEEE Computer Society, San Jose, California, USA, 553–567.

[10] Mark Burnett. 2015. Today I Am Releasing Ten Million Passwords. https://xato.net/today-i-am-releasing-ten-million-passwords-b6278bbe7495, as of September 10, 2018.

[11] William E. Burr, Donna F. Dodson, and W. Timothy Polk. 2004. Electronic Authentication Guideline: NIST SP 800-63 Ver. 1.0 (2004) to 800-63-2 (2013). https://csrc.nist.gov/publications/detail/sp/800-63/ver-10/archive/2004-06-30, as of September 10, 2018.

[12] Javier Carranza and Contributors. 2018. Ubiquity – Ubuntu Live CD Installer. https://launchpad.net/ubuntu/+source/ubiquity, as of September 10, 2018.

[13] Claude Castelluccia, Markus Dürmuth, and Daniele Perito. 2012. Adaptive Password-Strength Meters from Markov Models. In *Symposium on Network and Distributed System Security (NDSS '12)*. The Internet Society, San Diego, California, USA.

[14] Dashlane, Inc. 2018. Dashlane (Windows) – Password Manager. https://www.dashlane.com, as of September 10, 2018.

[15] "dcopi". 2013. NIST - Password Strength Meter Example. https://github.com/dcopi/PWStrength, as of September 10, 2018.

[16] Xavier de Carné de Carnavalet and Mohammad Mannan. 2014. From Very Weak to Very Strong: Analyzing Password-Strength Meters. In *Symposium on Network and Distributed System Security (NDSS '14)*. ISOC, San Diego, California, USA.

[17] Xavier de Carné de Carnavalet and Mohammad Mannan. 2014. Password Multi-Checker Tool. https://madiba.encs.concordia.ca/software/passwordchecker/, as of September 10, 2018.

[18] Matteo Dell'Amico and Maurizio Filippone. 2015. Monte Carlo Strength Evaluation: Fast and Reliable Password Checking. In *ACM Conference on Computer and Communications Security (CCS '15)*. ACM, Denver, Colorado, USA, 158–169.

[19] Matteo Dell'Amico, Pietro Michiardi, and Yves Roudier. 2010. Password Strength: An Empirical Analysis. In *Conference on Information Communications (INFOCOM '10)*. IEEE, San Diego, California, USA, 983–991.

[20] Dropbox, Inc. and Contributors. 2017. zxcvbn v4.4.2 – JavaScript Implementation of the zxcvbn Strength Meter. https://github.com/dropbox/zxcvbn, as of September 10, 2018.

[21] Markus Dürmuth, Fabian Angelstorf, Claude Castelluccia, Daniele Perito, and Abdelberi Chaabane. 2015. OMEN: Faster Password Guessing Using an Ordered Markov Enumerator. In *International Symposium on Engineering Secure Software and Systems (ESSoS '15)*. Springer, Milan, Italy, 119–132.

[22] Serge Egelman, Andreas Sotirakopoulos, Ildar Muslukhov, Konstantin Beznosov, and Cormac Herley. 2013. Does My Password Go Up to Eleven?: The Impact of Password Meters on Password Selection. In *ACM Conference on Human Factors in Computing Systems (CHI '13)*. ACM, Paris, France, 2379–2388.

[23] Dinei Florencio and Cormac Herley. 2007. A Large-scale Study of Web Password Habits. In *Conference on World Wide Web (WWW '07)*. ACM, Banff, Alberta, Canada, 657–666.

[24] Dinei Florêncio, Cormac Herley, and Paul C. van Oorschot. 2014. An Administrator's Guide to Internet Password Research. In *Large Installation System Administration Conference (LISA '14)*. USENIX, Seattle, Washington, USA, 44–61.

[25] Dinei Florêncio, Cormac Herley, and Paul C. van Oorschot. 2014. Password Portfolios and the Finite-Effort User: Sustainably Managing Large Numbers of Accounts. In *USENIX Security Symposium (SSYM '14)*. USENIX, San Diego, California, USA, 575–590.

[26] Dinei Florêncio, Cormac Herley, and Paul C. van Oorschot. 2016. Pushing on String: The "Don't Care" Region of Password Strength. *Commun. ACM* 59, 11 (Oct. 2016), 66–74.

[27] Maximilian Golla, Benedict Beuscher, and Markus Dürmuth. 2016. On the Security of Cracking-Resistant Password Vaults. In *ACM Conference on Computer and Communications Security (CCS '16)*. ACM, Vienna, Austria, 1230–1241.

[28] Maximilian Golla, Theodor Schnitzler, and Markus Dürmuth. 2018. "Will Any Password Do?" Exploring Rate-Limiting on the Web. In *Who Are You?! Adventures in Authentication Workshop (WAY '18)*. USENIX, Baltimore, Maryland, USA.

[29] Maximilian Golla, Ibrahim Sertkaya, and Markus Dürmuth. 2018. Password Strength Meter Comparison Website. https://password-meter-comparison.org, as of September 10, 2018.

[30] Jeremi M. Gosney. 2017. Nvidia GTX 1080 Ti Hashcat Benchmarks. https://gist.github.com/epixoip/ace60d09981be09544fdd35005051505, as of September 10, 2018.

[31] Paul A. Grassi, James L. Fenton, and William E. Burr. 2017. Digital Identity Guidelines – Authentication and Lifecycle Management: NIST SP 800-63B (2017).

[32] Yimin Guo and Zhenfeng Zhang. 2018. LPSE: Lightweight Password-Strength Estimation for Password Meters. *Computers & Security* 73 (March 2018), 507–518.

[33] Hana Habib, Jessica Colnago, William Melicher, Blase Ur, Sean Segreti, Lujo Bauer, Nicolas Christin, and Lorrie Cranor. 2017. Password Creation in the Presence

of Blacklists. In *Workshop on Usable Security (USEC '17)*. Internet Society, San Diego, California, USA.

[34] Shiva Houshmand and Sudhir Aggarwal. 2012. Building Better Passwords Using Probabilistic Techniques. In *Annual Computer Security Applications Conference (ACSAC '12)*. ACM, Orlando, Florida, USA, 109–118.

[35] Jason Huggins and SeleniumHQ Contributors. 2017. Selenium - Web Browser Automation. http://www.seleniumhq.org, as of September 10, 2018.

[36] Troy Hunt. 2018. 500m Pwned Passwords List. https://haveibeenpwned.com/Passwords, as of September 10, 2018.

[37] Markus Jakobsson and Mayank Dhiman. 2012. The Benefits of Understanding Passwords. In *USENIX Workshop on Hot Topics in Security (HotSec '12)*. USENIX, Bellevue, Washington, USA.

[38] JS Foundation. 2018. Appium – Automation Made Awesome. http://appium.io, as of September 10, 2018.

[39] Keeper Security, Inc. 2018. Keeper (Web) – Password Manager. https://keepersecurity.com, as of September 10, 2018.

[40] Patrick Kelley, Saranga Kom, Michelle L. Mazurek, Rich Shay, Tim Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio López. 2012. Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms. In *IEEE Symposium on Security and Privacy (SP '12)*. IEEE Computer Society, San Jose, California, USA, 523–537.

[41] Daniel V. Klein. 1990. "Foiling the Cracker": A Survey of, and Improvements to, Password Security. In *USENIX Security Workshop*. USENIX, Berkeley, California, USA, 5–14.

[42] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L. Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. 2011. Of Passwords and People: Measuring the Effect of Password-Composition Policies. In *ACM Conference on Human Factors in Computing Systems (CHI '11)*. ACM, Vancouver, British Columbia, Canada, 2595–2604.

[43] LogMeIn, Inc. 2018. LastPass (Web) – Password Manager. https://www.lastpass.com, as of September 10, 2018.

[44] Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. 2014. A Study of Probabilistic Password Models. In *IEEE Symposium on Security and Privacy (SP '14)*. IEEE, San Jose, CA, USA, 689–704.

[45] William Melicher. 2017. Source Code – Cracking Passwords with Neural Networks. https://github.com/cupslab/neural_network_cracking, as of September 10, 2018.

[46] 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 *USENIX Security Symposium (SSYM '16)*. USENIX, Austin, Texas, USA, 175–191.

[47] Robert Morris and Ken Thompson. 1979. Password Security: A Case History. *Commun. ACM* 22, 11 (1979), 594–597.

[48] Arvind Narayanan and Vitaly Shmatikov. 2005. Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff. In *ACM Conference on Computer and Communications Security (CCS '05)*. ACM, Alexandria, VA, USA, 364–372.

[49] Ronald Oussoren. 2018. PyObjC – The Python Objective-C Bridge. https://pythonhosted.org/pyobjc/, as of September 10, 2018.

[50] Sarah Pearman, Jeremy Thomas, Pardis Emami Naeini, Hana Habib, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman, and Alain Forget. 2017. Let's Go in for a Closer Look: Observing Passwords in Their Natural Habitat. In *ACM Conference on Computer and Communications Security (CCS '17)*. ACM, Dallas, Texas, USA, 295–310.

[51] Dominik Reichl. 2018. KeePass (Windows) – Password Manager. http://keepass.info/help/kb/pw_quality_est.html, as of September 10, 2018.

[52] Dominik Reichl. 2018. KPScript (Windows) – Scripting KeePass. http://keepass.info/help/v2_dev/scr_index.html, as of September 10, 2018.

[53] Stuart Schechter, Cormac Herley, and Michael Mitzenmacher. 2010. Popularity Is Everything: A New Approach to Protecting Passwords from Statistical-Guessing Attacks. In *USENIX Workshop on Hot Topics in Security (HotSec '10)*. USENIX, Washington, District of Columbia, USA.

[54] Sinew Software Systems. 2016. Enpass Release Notes – Use of the zxcvbn Strength Meter. https://www.enpass.io/release-notes/windowspc/, as of September 10, 2018.

[55] Sinew Software Systems. 2018. Enpass (Windows) – Password Manager. https://www.enpass.io, as of September 10, 2018.

[56] Eugene H. Spafford. 1992. Observing Reusable Password Choices. In *USENIX Security Symposium (SSYM '92)*. USENIX, Berkeley, California, USA, 299–312.

[57] Richard Tilley. 2018. Blooming Password. https://www.bloomingpassword.fun, as of September 10, 2018.

[58] Blase Ur. 2017. Source Code – Data-Driven Password Meter. https://github.com/cupslab/password_meter, as of September 10, 2018.

[59] Blase Ur, Felicia Alfieri, Maung Aung, Lujo Bauer, Nicolas Christin, Jessica Colnago, Lorrie Faith Cranor, Henry Dixon, Pardis Emami Naeini, Hana Habib, Noah Johnson, and William Melicher. 2017. Design and Evaluation of a Data-Driven Password Meter. In *ACM Conference on Human Factors in Computing Systems (CHI '17)*. ACM, Denver, Colorado, USA, 3775–3786.

[60] Blase Ur, Jonathan Bees, Sean M. Segreti, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2016. Do Users' Perceptions of Password Security Match Reality?. In *ACM Conference on Human Factors in Computing Systems (CHI '16)*. ACM, Santa Clara, California, USA, 3748–3760.

[61] Blase Ur, Patrick Gage Kelley, Saranga Komanduri, Joel Lee, Michael Maass, Michelle L. Mazurek, Timothy Passaro, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2012. How Does Your Password Measure Up? The Effect of Strength Meters on Password Creation. In *USENIX Security Symposium (SSYM '12)*. USENIX, Bellevue, Washington, USA, 65–80.

[62] 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 *USENIX Security Symposium (SSYM '15)*. USENIX, Washington, D.C., USA, 463–481.

[63] Ashlee Vance. 2010. If Your Password Is 123456, Just Make It HackMe. http://www.nytimes.com/2010/01/21/technology/21password.html, as of September 10, 2018.

[64] Rafael Veras, Christopher Collins, and Julie Thorpe. 2014. On the Semantic Patterns of Passwords and their Security Impact. In *Symposium on Network and Distributed System Security (NDSS '14)*. The Internet Society, San Diego, California, USA.

[65] Ding Wang, Debiao He, Haibo Cheng, and Ping Wang. 2016. fuzzyPSM: A New Password Strength Meter Using Fuzzy Probabilistic Context-Free Grammars. In *Conference on Dependable Systems and Networks (DSN '16)*. IEEE Computer Society, Toulouse, France, 595–606.

[66] Ding Wang, Zijian Zhang, Ping Wang, Jeff Yan, and Xinyi Huang. 2016. Targeted Online Password Guessing: An Underestimated Threat. In *ACM Conference on Computer and Communications Security (CCS '16)*. ACM, Vienna, Austria, 1242–1254.

[67] Miranda Wei, Maximilian Golla, and Blase Ur. 2018. The Password Doesn't Fall Far: How Service Influences Password Choice. In *Who Are You?! Adventures in Authentication Workshop (WAY '18)*. USENIX, Baltimore, Maryland, USA.

[68] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. 2010. Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords. In *ACM Conference on Computer and Communications Security (CCS '10)*. ACM, Chicago, Illinois, USA, 162–175.

[69] Matt Weir, Sudhir Aggarwal, Breno de Medeiros, and Bill Glodek. 2009. Password Cracking Using Probabilistic Context-Free Grammars. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, Oakland, California, USA, 391–405.

[70] Matt Weir, Sudhir Aggarwal, Breno De Medeiros, and Bill Glodek. 2009. Password Cracking Using Probabilistic Context-Free Grammars. In *IEEE Symposium on Security and Privacy (SP '09)*. IEEE, Berkeley, CA, USA, 391–405.

[71] Daniel Lowe Wheeler. 2016. zxcvbn: Low-Budget Password Strength Estimation. In *USENIX Security Symposium (SSYM '16)*. USENIX, Austin, Texas, USA, 157–173.

A METER COMPARISON

In the following, we list the full results of our data collection. We separated the five categories *Academic Proposals*, *Password Managers*, *Operating Systems*, *Websites*, and *Previous Work* into two tables. A colorful version that allows easier comparison can be found online [29].

A.1 Academic Proposals, Password Managers, and Operating Systems

Table 5: We computed the weighted Spearman correlation as the best similarity score (cf. Section 5). The table lists the online use case on the left, the offline use case on the right. We highlighted if and on how many bins a meter quantized its output. Additionally, we list whether a meter runs on client- or server-side and how the meter visualizes the strength to the user.

					Online Attacker			Offline Attacker		
ID	**Meter**	**Type**	**Quant.**	**Visu.**	**RockYou**	**LinkedIn**	**000Webhost**	**RockYou**	**LinkedIn**	**000Webhost**
					Academic Proposals					
1A	**Comprehensive8** [61]	C	-	-	-0.652	-0.589	0.251	-0.476	-0.616	0.441
1B	**Comprehensive8** [61]	C	Q5	Text	-0.331	-0.084	0.409	-0.128	-0.123	0.421
2	**Eleven** [22]	C	-	-	0.670	0.912	0.492	0.755	0.951	0.733
3	**LPSE** [32]	C	Q3	-	0.584	0.669	0.508	0.544	0.718	0.693
4A	**Markov (OMEN)** [13]	S	-	-	0.721	0.697	0.410	0.701	0.669	0.660
4B	**Markov (Single)** [27]	S	-	-	0.718	0.998	0.817	0.828	0.991	0.872
4C	**Markov (Multi)** [27]	S	-	-	0.721	0.998	0.902	0.997	0.995	0.777
5A	**NIST** [11]	C	-	-	0.670	0.912	0.492	0.755	0.951	0.733
5B	**NIST (w. Dict.)** [11]	C	-	-	0.669	0.910	0.472	0.756	0.953	0.816
6	**PCFG (fuzzyPSM)** [65]	S	-	-	1.000	0.994	0.963	0.998	0.999	0.899
7A	**RNN Generic** [46]	C	-	-	0.632	0.542	0.427	0.535	0.520	0.800
7B	**RNN Generic (Web)** [59]	C	-	-	0.473	0.649	0.421	0.449	0.688	0.777
7C	**RNN Target** [46]	C	-	-	0.951	0.913	0.965	0.896	0.860	0.885
7D	**RNN Target (w. Bloom)** [46]	C	-	-	0.951	0.913	0.965	0.896	0.860	0.882
8A	**zxcvbn (Guess Number)** [71]	C	-	-	0.989	0.990	0.554	0.989	0.999	0.868
8B	**zxcvbn (Score)** [71]	C	Q5	-	0.341	0.490	0.359	0.373	0.567	0.817
					Password Managers					
9A	**1Password (Web)**	C	-	Bar	0.276	0.433	0.441	0.401	0.621	0.807
9B	**1Password (Web)**	C	Q5	Text (Int.)	0.276	0.433	0.407	0.401	0.621	0.813
10A	**Bitwarden (Web)**	C	-	Bar	-0.635	-0.490	0.418	-0.457	-0.540	0.676
10B	**Bitwarden (Web)**	C	Q3	Text (Int.)	0.258	0.372	0.494	0.333	0.340	0.725
11	**Dashlane 5.5 (Windows)**	C	-	Text	0.686	0.785	0.241	0.698	0.820	0.410
12	**Enpass 5.6.8 (Windows)**[5]	C	Q5	Bar a. Text [Z]	0.341	0.490	0.359	0.373	0.567	0.817
13A	**KeePass 2.38 (Windows)**	C	-	Bar	0.856	0.785	0.393	0.884	0.870	0.744
13B	**KeePass 2.38 (Windows)**	C	Q5	Text	0.000	0.000	0.045	0.003	0.002	0.321
14A	**Keeper (Web)**	C	Q5	Bar	0.200	0.258	0.400	0.223	0.238	0.589
14B	**Keeper (Web)**	C	-	Score (Int.)	0.805	0.719	0.284	0.869	0.824	0.476
15	**LastPass (Web)**	C	Q5	Bar [Z]	0.197	0.428	0.266	0.232	0.510	0.717
16A	**LogMeOnce (Web)**	C	-	Bar	0.425	0.559	0.245	0.410	0.602	0.503
16B	**LogMeOnce (Web)**	C	Q5	Text	0.053	0.138	0.315	0.070	0.130	0.541
17A	**RoboForm 8.4.8.8 (Chrome)**	C	Q4	Text [Z]	0.740	0.773	0.477	0.711	0.827	0.759
17B	**RoboForm 8.4.8.8 (Chrome)**	C	-	Score (Int.) [Z]	0.685	0.932	0.528	0.781	0.962	0.725
17C	**RoboForm Business (Web)**	C	Q6	Text [Z]	0.523	0.693	0.402	0.553	0.727	0.738
18	**True Key 2.8.5711 (Chrome)**	C	Q5	Text [Z]	0.341	0.490	0.359	0.373	0.567	0.817
19A	**Zoho Vault (Web)**	C	Q3	Bar a. Text	0.088	0.134	0.120	0.104	0.107	0.506
19B	**Zoho Vault (Web)**	C	-	Score (Int.)	0.464	0.502	0.509	0.450	0.468	0.727
					Operating Systems					
20A	**macOS High Sierra 10.13.4**	C	-	Bar	-0.667	-0.513	0.450	-0.488	-0.569	0.726
20B	**macOS High Sierra 10.13.4**	C	Q4	Text	0.072	0.204	0.469	0.094	0.171	0.728
20C	**macOS High Sierra 10.13.4**	C	-	Bar (Hover)	-0.667	-0.513	0.449	-0.488	-0.569	0.727
21A	**Ubuntu 18.04 (Ubiquity)**	C	Q5	Text	-0.849	-0.808	-0.141	-0.792	-0.851	0.132
21B	**Ubuntu 18.04 (Ubiquity)**	C	-	Score (Int.)	-0.818	-0.817	-0.189	-0.779	-0.855	0.002

Type: C=Client, S=Server, H=Hybrid;
Quantization: Q3–Q6=Number of bins, e. g., Q5=[`Terrible, Weak, Good, Excellent, Fantastic`];
Visualization: Bar=Bar-based meter, Text=Textual strength description, (Int.)=Internal value not displayed, [Z]=*zxcvbn*-based meter.

[5]Not crawled, analysis showed the use of plain zxcvbn (Score) MID 8B.

A.2 Websites and Comparison with Previous Work

Table 6: We computed the weighted Spearman correlation as the best similarity score (cf. Section 5). The table lists the online use case on the left, the offline use case on the right. We highlighted if and on how many bins a meter quantized its output. Additionally, we list whether a meter runs on client- or server-side and how the meter visualizes the strength to the user.

ID	Meter	Type	Quant.	Visu.	Online Attacker			Offline Attacker		
					RockYou	LinkedIn	000Webhost	RockYou	LinkedIn	000Webhost
					Websites					
22	**Airbnb**	C	Q3	Text	0.054	0.113	0.331	0.063	0.141	0.605
23A	**Apple**	H	Q4	Bar	0.000	0.020	0.102	0.000	0.046	0.345
23B	**Apple**	H	Q3	Text	0.000	0.020	0.102	0.000	0.046	0.345
24	**Baidu**	S	Q3	Text	0.829	0.828	0.154	0.825	0.875	0.350
25A	**Best Buy**	C	-	Bar	0.676	0.912	0.424	0.765	0.949	0.645
25B	**Best Buy**	C	Q3	Text	0.074	0.102	0.331	0.077	0.095	0.511
26A	**China Railway (12306.cn)**	C	Q3	Bar	0.161	0.226	0.346	0.166	0.197	0.571
27A	**Dropbox**	C	Q4	Bar [Z]	0.056	0.094	0.087	0.076	0.108	0.611
28A	**Drupal 8.5.3**	C	-	Bar	0.677	0.788	0.490	0.688	0.822	0.732
28B	**Drupal 8.5.3**	C	Q4	Text	0.022	0.019	0.157	0.022	0.039	0.356
29	**eBay (PW Change)**	H	Q4	Bar	0.031	0.120	-0.373	0.157	0.081	-0.146
30	**Facebook (PW Change)**	C	Q4	Text	-0.066	0.372	0.498	0.118	0.339	0.725
31A	**FedEx**	C	Q3	Bar	0.000	0.090	0.147	0.007	0.071	0.345
31B	**FedEx**	C	Q5	Score (Int.)	0.000	0.090	0.147	0.007	0.071	0.345
32A	**Google**	H	Q5	Bar	0.522	0.692	0.586	0.551	0.729	0.763
33	**Have I Been Pwned?**	S	-	Text	0.992	0.996	0.739	0.991	0.997	0.939
34	**The Home Depot**	C	Q3	Bar a. Text	0.362	0.548	0.475	0.420	0.604	0.731
35A	**Microsoft (v3)**[6]	C	Q4	Bar	0.521	0.694	0.487	0.551	0.726	0.724
35B	**Microsoft (v3)**[6]	C	-	Score (Int.)	0.670	0.912	0.491	0.755	0.951	0.734
36	**reddit**	C	Q5	Bar [Z]	0.341	0.490	0.359	0.373	0.567	0.817
37	**Sony Entertainment Network**	C	Q3	Bar	0.115	0.185	0.188	0.128	0.169	0.511
38	**Sina Weibo**	C	Q4	Text	0.427	0.779	0.544	0.500	0.803	0.502
39	**Tumblr**	S	Q6	Bar	0.499	0.576	0.165	0.550	0.514	0.394
40A	**Twitch**	C	Q5	Bar [Z]	0.197	0.428	0.266	0.232	0.510	0.717
40B	**Twitch**	C	Q3	Text [Z]	0.197	0.427	0.300	0.232	0.510	0.712
41A	**Twitter**	H	-	Bar	0.643	0.637	0.509	0.581	0.674	0.769
41B	**Twitter**	H	Q5	Score (Int.)	0.554	0.629	0.585	0.526	0.665	0.681
42A	**Yandex**	H	-	Bar	-0.392	-0.082	0.494	-0.224	-0.117	0.733
42B	**Yandex**	H	Q4	Text	0.370	0.724	0.502	0.475	0.775	0.799
					de Carné de Carnavalet and Mannan (2014) [16]					
23C	**Apple**	H	Q4	-	0.521	0.694	0.462	0.551	0.726	0.707
26B	**China Railway (12306.cn)**	C	Q3	-	0.160	0.226	0.346	0.165	0.197	0.571
27B	**Dropbox**	C	Q5	- [Z]	0.085	0.104	0.121	0.121	0.131	0.654
28C	**Drupal**	C	Q4	-	-0.187	0.256	0.148	-0.095	0.233	0.350
31C	**Fedex**	C	Q5	-	0.000	0.090	0.147	0.007	0.071	0.345
32B	**Google**	S	Q5	-	0.521	0.694	0.507	0.551	0.726	0.717
35C	**Microsoft (v3)**	C	Q4	-	0.521	0.694	0.487	0.551	0.726	0.724
43	**PayPal**	H	Q4	-	0.521	0.694	0.444	0.552	0.727	0.736
44	**QQ**	C	Q4	-	0.844	0.874	0.492	0.867	0.918	0.721
41C	**Twitter**	C	Q6	-	0.223	0.638	0.514	0.271	0.674	0.658
45	**Yahoo!**	C	Q4	-	-0.187	0.256	0.142	-0.095	0.233	0.346
42C	**Yandex**	S	Q4	-	0.150	0.581	0.398	0.217	0.624	0.709

Type: C=Client, S=Server, H=Hybrid;
Quantization: Q3–Q6=Number of bins, e. g., Q5=[`Terrible, Weak, Good, Excellent, Fantastic`];
Visualization: Bar=Bar-based meter, Text=Textual strength description, (Int.)=Internal value not displayed, [Z]=*zxcvbn*-based meter.

[6] MID 35A/35B have been deprecated by Microsoft in 2016.