No Single Silver Bullet:
Measuring the Accuracy of Password Strength Meters

Ding Wang, Xuan Shan, Qiying Dong
Nankai University
{wangding, shanxuan, dqy}@nankai.edu.cn

Yaosheng Shen
Peking University
ysshen@pku.edu.cn

Chunfu Jia
Nankai University
cfjia@nankai.edu.cn

Abstract

To help users create stronger passwords, nearly every respectable web service adopts a password strength meter (PSM) to provide real-time strength feedback upon user registration and password change. Recent research has found that PSMs that provide accurate feedback can indeed effectively nudge users toward choosing stronger passwords. Thus, it is imperative to systematically evaluate existing PSMs to facilitate the selection of accurate ones. In this paper, we highlight that *there is no single silver bullet metric* for measuring the accuracy of PSMs: For each given guessing scenario and strategy, a specific metric is necessary. We investigate the intrinsic characteristics of online and offline guessing scenarios, and *for the first time*, propose a *systematic evaluation framework* that is composed of four different dimensioned criteria to rate PSM accuracy under these two password guessing scenarios (as well as various guessing strategies).

More specifically, for *online guessing*, the strength misjudgments of passwords with different popularity would have varied effects on PSM accuracy, and we suggest the weighted Spearman metric and consider two typical attackers: The general attacker who is unaware of the target password distribution, and the knowledgeable attacker aware of it. For *offline guessing*, since the cracked passwords are generally weaker than the uncracked ones, and they correspond to two disparate distributions, we adopt the Kullback-Leibler divergence metric and investigate the four most typical guessing strategies: brute-force, dictionary-based, probability-based, and a combination of above three strategies. In particular, we propose the Precision metric to measure PSM accuracy when non-binned strength feedback (e.g., probability) is transformed into easy-to-understand bins/scores (e.g., [weak, medium, strong]). We further introduce a reconciled Precision metric to characterize the impacts of strength misjudgments in different directions (e.g., weak$\rightarrow$strong and strong$\rightarrow$weak) on PSM accuracy. The effectiveness and practicality of our evaluation framework are demonstrated by rating 12 leading PSMs, leveraging 14 real-world password datasets. Finally, we provide three recommendations to help improve the accuracy of PSMs.

1 Introduction

Passwords firmly remain the dominant mechanism for user access control on the Web, ranging from low-value news portals [1], moderate-value e-commerce and email services [2] to highly-sensitive genomic data protection [3]. It is well known that users tend to choose weak passwords which are prone to guessing attacks [4, 5]. To deal with such vulnerable behaviors, almost every respectable web service now employs a password strength meter (PSM) to encourage users toward strong passwords, see Fig. 1 for examples (i.e., 12306-PSM and Google-PSM). PSMs with accurate strength feedback can indeed play a positive role in the password creation stage, but inaccurate PSMs do more harm than good [6–9]: If a weak password is rated as strong, it will give users an unrealistic sense of security and compromise password security [10, 11]; Whereas, if a strong password is evaluated as weak, users need to spend longer creating their passwords and are prone to feel frustrated or annoyed [6, 8].

Over the past decades, academia and industry have made impressive efforts to design PSMs (e.g., rule-based [12, 13], pattern-based [14, 15], and guessability-based [16–20]) with accurate strength feedback. Although dozens of PSMs have been proposed one after another, it is unclear which one is most suitable for which scenario, especially when there are various guessing attacks (e.g., online guessing and offline guessing) and application scenarios (e.g., e-banking and cloud). To answer this question, a more fundamental issue needs to be first addressed: *how to measure the accuracy of PSMs.* At NDSS'12, Castelluccia et al. [16] proposed a new PSM and the first metric (i.e., the Spearman rank correlation) to measure PSM accuracy, but they only focused on the online guessing scenario. At IEEE TIFS'17, Galbally et al. [19] proposed a new multimodal-based PSM, and suggested using the *single* metric Kullback-Leibler divergence to measure PSM accuracy, but they only considered the offline guessing scenario. Like these two works, most existing works (e.g., [13, 14, 17]) present new PSMs with assertions of the superior aspects over previous ones, while (unconsciously) overlooking dimensions on which their schemes fare poorly.

1.1 Motivations

Accuracy is the most essential property of a PSM, yet few studies have paid attention to the fundamental accuracy evaluation criteria framework that allows for a *systematic* and *thorough* assessment of the accuracy of existing (highly heterogeneous) PSMs. Without a systematic methodology, a fair comparison is unlikely, and there is no basis for diversified PSM schemes to be rated across a common spectrum. As far as we know, Golla and Dürmuth's work [10] at CCS'18 may be the closest to our paper: They proposed a *single* metric (i.e., weighted Spearman correlation coefficient, which is only applicable to online guessing scenarios) for measuring PSM accuracy, and compared 45 PSMs on only three *English* password datasets. In a nutshell, little existing work pays attention to the evaluation framework for PSMs, and *the following key research questions (**RQs**) remain to be answered:*

RQ1: Though the characteristics of online and offline guessing scenarios are vastly different (e.g., allowed guessing attempts), whether PSMs can be evaluated by using the same criterion under these two guessing scenarios?

RQ2: As attackers in different guessing scenarios have varied knowledge of the targets and suffers from different constraints (and thus adopts different attacking strategies), should specific evaluation metrics be selected accordingly? Or, is there a single silver bullet metric?

RQ3: Since the strength misjudgments in different directions (e.g., weak$\rightarrow$strong vs. strong$\rightarrow$weak) have varied impacts on PSM accuracy, how should the evaluation metric be adjusted accordingly?

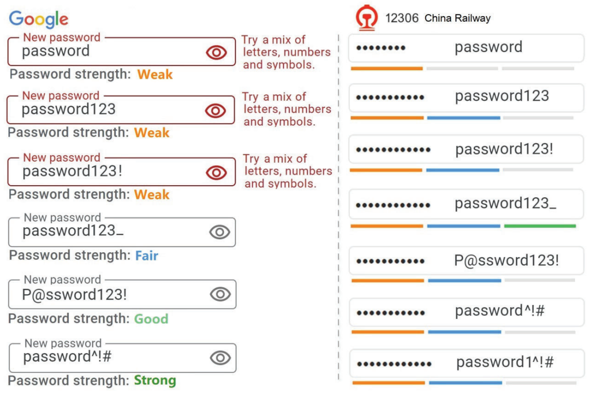
RQ4: How will the accuracy of a PSM change when it is deployed on sites of a different language from which it is originally designed for? For instance, how will the widely adopted Zxcvbn [14] perform on Chinese sites, while it is primarily designed for English sites?

RQ5: How will the accuracy of a PSM change when its non-binned strength feedback (e.g., entropy or guess number) is converted to easy-to-understand (and widely used) bins/scores (e.g., [weak, medium, strong])?

It is imperative to address the above research questions to facilitate the design, evaluation and selection of a proper PSM for a given site, and *our work takes the first step towards addressing these questions*.

1.2 Contributions

We first investigate the intrinsic characteristics of online and offline guessing scenarios (and various guessing strategies), and explicitly reveal that "***there is no single silver bullet metric***" for evaluating PSM accuracy. Accordingly, we, *for the first time*, propose a *systematic evaluation framework* composed of four different dimensioned criteria to rate PSM accuracy under both guessing scenarios (and various guessing strategies). The effectiveness and practicality of this framework are demonstrated by rating 12 leading PSMs, leveraging


Google

New password
password
Password strength: Weak
Try a mix of letters, numbers and symbols.

New password
password123
Password strength: Weak
Try a mix of letters, numbers and symbols.

New password
password123!
Password strength: Weak
Try a mix of letters, numbers and symbols.

New password
password123_
Password strength: Fair

New password
P@ssword123!
Password strength: Good

New password
password^!#
Password strength: Strong

12306 China Railway

•••••••• password
••••••••••• password123
•••••••••••• password123!
•••••••••••• password123_
••••••••••• P@ssword123!
••••••••••• password^!#
•••••••••••• password1^!#

Figure 1: Inaccurate (see `P@ssword123!`) and inconsistent (see `password 123_`) PSM feedback: Google (`https://myaccount.google.com/`) and China Railway 12306 (`https://www.12306.cn/en/register.html`).

218.79 million passwords from 14 high-profile English and Chinese sites. Note that our evaluation framework can also be applied to *targeted* online/offline scenarios with minor adjustments. Though not cast in stone, our framework is expected to help better evaluate current/future PSMs, and to inform future PSMs designs. In summary, our contributions are:

- **Metrics for online guessing.** While the strength rankings of popular passwords can be approximated by their frequency rankings, the misjudgment of the strength of passwords with different popularity would have varied effects on PSM accuracy. Thus, we use the weighted Spearman correlation coefficient (WSpearman) metric for online guessing scenarios, and investigate the two most typical online attackers: 1) The general attacker, who is unaware of the target distribution and will mainly traverse common popular passwords [4]; and 2) The knowledgeable attacker, who is well-informed of the target password distribution and will give priority to these most popular passwords (e.g., top-10^4) [4, 21, 22]. We also investigate and compare the characteristics of six WSpearman calculation methods and adopt the best one.

- **Metrics for offline guessing.** We consider the four most typical offline guessing attackers: Brute-force, dictionary-based, probability-based, and the combined one with the aforementioned three attacking strategies [23, 24]. We use the Kullback-Leibler divergence metric to measure the differences between the strength distributions of the cracked passwords and the remaining (uncracked) passwords. In particular, we propose a new metric, namely Precision, to measure the accuracy of a PSM when its non-binned strength feedback (e.g., entropy or guess number) is transformed into easy-to-understand (and widely used) bins/scores (e.g., [weak, medium, strong]). Further, we design a reconciled Precision metric (i.e., Precision^{Security}) that can characterize the strength misjudgments in different directions (e.g., weak$\rightarrow$strong and strong$\rightarrow$weak) on PSM accuracy.

- **Extensive evaluation.** We utilize our evaluation framework to systematically measure 12 state-of-the-art PSMs: seven foremost academic PSMs (i.e., fuzzyPSM [17], MultiPSM [19], PCFG-PSM [25], Markov-PSM [16], RNN-PSM [18], LPSE [13] and CNN-PSM [20]), four representative commercial PSMs (i.e., Zxcvbn [14], KeePSM [15], 12306-PSM and Microsoft-PSM [26]), and our modified Zxcvbn$_C$. We conduct experiments on 139.71 million passwords from seven English sites and 79.08 million passwords from seven Chinese sites, filling the gap in [10,27] which only focused on how PSMs perform on English passwords. Results show that, in online guessing scenarios, fuzzyPSM [17] is the most accurate PSM, and pattern detection-based PSMs (especially Zxcvbn [14]) are more accurate than attack algorithm-based PSMs. In offline scenarios, multimodal-based MultiPSM [23] performs the best, and Markov-PSM [16] and Zxcvbn [14] also show good accuracy.
- **Some insights.** We gain a number of insights from our experiments and provide a few suggestions to help improve PSMs. We find that, adaptive score conversion methods can be used to facilitate PSM application. Besides offline scenarios, PSMs can also be effectively integrated to perform better in online scenarios. Through cross-language evaluations, we also provide the first concrete evidence from extensive empirical experiments that PSMs (and their training sets) need to be adapted to accommodate different languages/services (e.g., we modify the English-originated Zxcvbn [14] to construct a more accurate Zxcvbn$_C$ for Chinese sites).

2 Related work

Early password-strength meters (PSMs) are dated back to the 1990s, and they are named proactive password checkers (see [28,29]). Their basic idea is to check each user-chosen password candidate against a dictionary of weak passwords. However, Yan [30] pointed out that these checkers often miss weak passwords (e.g., `12a34b5`) that do not fall into the dictionary. Thus, he proposed to adopt entropy-based checking as a complementary method to improve dictionary-based checking. In 2006, the influential NIST authentication guideline [12] followed this idea and suggested a password strength meter (PSM) based on heuristic estimations of password entropy (e.g., "a bonus of 6 bits of entropy is assigned for a composition rule that requires both upper case and non-alphabetic characters" and "a bonus of up to 6 bits of entropy is added for an extensive dictionary check"). Because such heuristics are too coarse to capture users' highly predictable tricks to improve password complexity, entropy is unsuitable for assessing user-generated password strength.

Although NIST-PSM [12] has been abandoned by the latest version of NIST SP800-63B since 2017 [31], most high-profile services (e.g., Google and 12306; see Fig. 1) are still following the entropy idea of NIST-PSM [2,27,32]. We take Google-PSM as a concrete example. The strength feedback of Google-PSM is an ad-hoc combination of password length, character types used, and dictionary checking (see Sec. V-B of [27] for details). Although their design ideas are similar, the strength feedback of different sites' meters has a large variance. It has been shown that meters of different sites produce highly inconsistent results when checking the same password [27,32] (see Fig. 1 for examples). Inaccurate and misleading feedback would not only impair security, but also confuse/frustrate users, defeating the purpose of PSMs [27].

The heuristic entropy metric has been replaced by "guessability", because the formers on the basis of some simple rules have been reported are insufficiently reflecting the resistance of passwords to advanced guessing attacks [33, 34]. Password guessability measures the ability to withstand guessing by a particular password cracker, which is generally instantiated by the guess number, the time required for guessing, and so on. However, it is usually expensive to compute the exact guess number of an unpopular password. At CCS'15, Dell'Amico and Filippone [35] introduced the Monte Carlo sampling method to efficiently and accurately estimate the guess number of a password. This method applies to a broad set of probabilistic password models such as probabilistic context-free grammars (PCFG) [36] and Markov chains [37], making the guess number a feasible metric in reality.

All these bring out a number of recent academic PSMs (see more information in Table 3) that are based on "guessability", and they all show advantages over existing entropy-based PSMs (e.g., Microsoft-PSM [26]). For example, Markov-PSM [16] has a higher Spearman correlation coefficient than the NIST-PSM [12] and its variations (e.g., Google-PSM and Microsoft-PSM), when an ideal PSM is adopted as the benchmark for strength reference; Wang et al. [17] employed both the Spearman and Kendall correlation coefficients to empirically demonstrate the superiority of fuzzyPSM [17] over Markov-PSM [16], Zxcvbn [14] and PCFG-PSM [25], when considering online guessing attacks (i.e., the guess number allowed is small, usually$<10^4$); Melicher et al. [18] compared the number of "unsafe errors" (caused by overestimating the Password strength) produced by RNN-PSM [18], Zxcvbn [14] and Yahoo-PSM, indicating that RNN-PSM [18] is more accurate due to fewer unsafe errors. Pasquini et al. [20] proposed a PSM with character-level feedback based on Convolutional Neural Network (denoted as CNN-PSM), and it was shown to be more accurate than RNN-PSM [18] and Markov-PSM [16].

A significant drawback of guessability-based PSMs is that (simulated) guessing algorithms may not be as effective as those adopted by real attackers, thus introducing a strong bias in the guessability/strength estimation [11,38]. In 2017, Galbally et al. [19] pointed out that no existing PSM can be applied to all guessing strategies. In response, they presented a multimodal-based MultiPSM, whose output is a fusion score that heuristically combines four heterogeneous

Table 1: Basic information about trawling guessing scenarios.

Guessing scenarios	Guessing strategies	Allowed guesses	Interact with server	Main constraint	Efficacy metric
Online [33, 39, 42]	Popularity-based	$\leq 10^{4\ddagger}$	Yes	Detection, lockout	Cracked%, # of guesses needed
Offline [36, 37, 43]	Brute-force	$>10^9$	No	Attacker power	Cracked%, # of guesses needed, Attacking time†
	Dictionary-based	$>10^9$			
	Probability-based	$>10^9$			

†Attacking time = guesses needed $\times$ attacker power (i.e., how many times of password guesses can be computed per second).

‡ We provide the rationality of selecting 10^4 as the online threshold in Appendix A.

strength scores to benefit from each guessing model's advantages (see Sec. 4.2 for more details). Their results show that, in terms of Kullback-Leibler divergence, MultiPSM [19] outperforms NIST-PSM [12], Yahoo-PSM and Google-PSM in the brute-force attack, dictionary-based attack and dictionary with rules attack scenarios. Despite this, it is still unknown the performance of MultiPSM [19] in more effective probability guessing scenarios (e.g., Markov [37]). Similarly, it is also unknown whether MultiPSM [19] can provide accurate strength evaluation results in online guessing (the primary threat that users need to mitigate [39, 40]).

3 Systematic evaluation framework

3.1 Architecture and adversary model

Without loss of generality, we consider password-based authentication with a client-server architecture. Each user U first registers at the server S, and S keeps U's password (in a salted hash as recommended [31]). When the underlying authentication protocol (e.g., OPAQUE [41]) is secure, the adversary $\mathcal{A}$ has to obtain U's password to break U's account (e.g., through guessing, keylogging and shoulder surfing). In this work, we mainly explore the trawling guessing attacks for three reasons: (1) Other threats (e.g., keylogging) are unrelated to password strength; (2) Mainstream PSMs (e.g., [14, 16–19, 25]) are designed to help users generate secure passwords that can withstand trawling guessing attacks; (3) Due to the sensitive nature of personal data, it is highly controversial for sites to collect and maintain the user's personal information (and passwords leaked from other sites). Trawling guessing attacks aim to crack as many passwords as possible, not to crack the password of a specific user. According to where the attack is performed, trawling guessing scenarios can be divided into online and offline attacks. The basic information of the two kinds of attacks is shown in Table 1.

In online guessing scenarios, the attacker $\mathcal{A}$ checks her guesses by constantly attempting to login as legitimate users, while an offline attacker checks her guesses by searching for collisions of password hashes on the hardware she controls. Therefore, a well-designed server S can reduce $\mathcal{A}$'s advantage by limiting the login rate [5, 44] (see Appendix A for more details), forcing users to reset their passwords when the failed attempts exceed a threshold [44], and activating multi-factor authentication mechanisms. On the contrary, in offline guessing scenarios, the attacker $\mathcal{A}$ has already obtained all the data that can be used to *locally* verify her guesses: It is usually a salt-hashed password file obtained through database leaks. Here, rate-limiting (at the server side) is inapplicable, and $\mathcal{A}$ is only constrained by her local computing resources and the time allowed [17, 40, 45].

Different application scenarios may suffer from different kinds of guessing attacks. For instance, e-banking is more prone to *online* guessing attacks than *offline* attacks, because the backend bank server is generally better guarded than common sites and its password files are less likely to be leaked; In contrast, portals like Yahoo are more likely to suffer *offline* guessing attacks, because their backend servers/systems are so complex that it is virtually impossible to keep them safe. Actually, Yahoo has suffered at least three major user account data breaches, with more than 3 billion victims [46].

For offline guessing, traditional heuristic attacks mainly include brute-force and dictionary-based attacks, and they have been adopted by automatic password-cracking tools such as John the Ripper (JtR) [47] and HashCat [48]. In 2009, Weir et al. [36] proposed a seminal probability-based guessing algorithm, and it can crack 28%∼129% more passwords than JtR [47]. Since then, several probability-based guessing algorithms (e.g., [18, 21, 24, 37]) have been proposed, which have higher cracking rates than traditional heuristic attacks. However, these algorithms cannot substitute traditional (brute-force and dictionary-based) attacks because the latter are faster, cheaper and easier to deploy [11, 38, 49]. Thus, a powerful realistic attacker is likely to adopt both traditional and advanced probability-based attacks.

As the attacker in different guessing scenarios suffers from essentially different constraints, the focus of a PSM should also differ accordingly. In the online guessing scenario where the number of guesses is limited, a PSM should accurately detect *popular passwords* (also the preferred guessing passwords of attackers), especially the most popular passwords (e.g., `123456`). In offline guessing scenarios, a PSM needs to prevent *easy-to-crack passwords* (including but not limited to popular passwords and their simple variants) from registering as much as possible, reducing the number of passwords an attacker can successfully crack within given computing resources. As different guessing scenarios entail the attacker $\mathcal{A}$ with different abilities and resources, a sound PSM should reflect the extent to which a password would consume $\mathcal{A}$'s resources in the given scenario. This implies that the accuracy of PSMs should be evaluated with different criteria in different guessing scenarios, giving **RQ1** a *negative* answer.

3.2 Guessing strategies

Since a realistic attacker would implement guessing in line with her abilities and resources [39], we consider the following most typical attackers with specific guessing strategies:

In *online guessing*, an attacker can only perform very limited guessing attempts due to the protection measures (e.g., risk-based account lockout [50]) deployed on the server. Since the protection measures are quite diverse, the exact value of

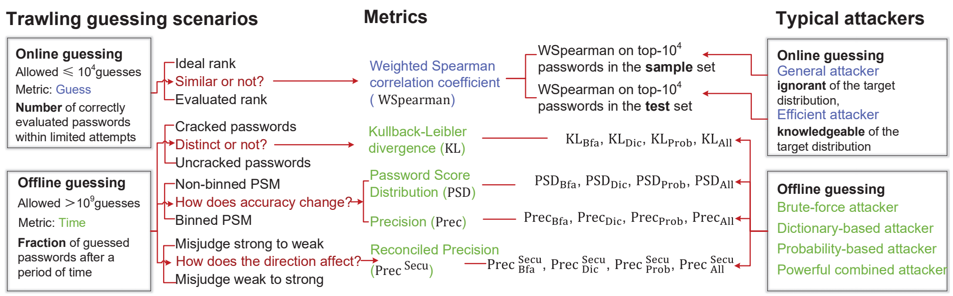

Trawling guessing scenarios
Online guessing
Allowed ⩽ 10^4 guesses
Metric: Guess
Number of correctly evaluated passwords within limited attempts
Offline guessing
Allowed > 10^9 guesses
Metric: Time
Fraction of guessed passwords after a period of time
Ideal rank
Similar or not?
Evaluated rank
Cracked passwords
Distinct or not?
Uncracked passwords
Non-binned PSM
How does accuracy change?
Binned PSM
Misjudge strong to weak
How does the direction affect?
Misjudge weak to strong
Metrics
Weighted Spearman correlation coefficient (WSpearman)
Kullback-Leibler divergence (KL)
Password Score Distribution (PSD)
Precision (Prec)
Reconciled Precision (Prec^Secu)
WSpearman on top-10^4 passwords in the sample set
WSpearman on top-10^4 passwords in the test set
KL_Bfa, KL_Dic, KL_Prob, KL_All
PSD_Bfa, PSD_Dic, PSD_Prob, PSD_All
Prec_Bfa, Prec_Dic, Prec_Prob, Prec_All
Prec^Secu_Bfa, Prec^Secu_Dic, Prec^Secu_Prob, Prec^Secu_All
Typical attackers
Online guessing
General attacker
ignorant of the target distribution,
Efficient attacker
knowledgeable of the target distribution
Offline guessing
Brute-force attacker
Dictionary-based attacker
Probability-based attacker
Powerful combined attacker

Figure 2: Our proposed systematic evaluation framework for measuring PSM accuracy.

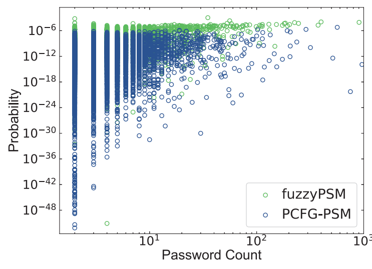


Figure 3: PSM scatter plot of 10,000 passwords randomly selected from 000webhost.

the online guessing threshold $\mathcal{T}$ depends on the target system's risk analysis results. Without loss of generality, we set $\mathcal{T}=10^4$ according to our manual test (see Appendix A) and rule-of-thumb recommendations in [17, 39, 40, 51]. Given $\mathcal{T}$, for an attacker well-informed of the target password distribution, her sensible strategy is to try the most popular passwords (e.g., top-10^4) in the known distribution [4]. We call her a *knowledgeable* attacker. In contrast, for an attacker unaware of the target distribution, her sensible strategy is to traverse *common* popular passwords [4, 21] (e.g., constructed from leaked password corpus), and we call her a *general* attacker.

In *offline guessing* scenarios, it is not difficult for attackers to make more than billions of guesses by using GPUs [52,53]. Generally, offline password attacks fall into three main groups: brute-force, dictionary-based and probability-based attacks. A *brute-force* attacker performs an exhaustive search over all possible passwords in a given search space, and she will soon reach the limit of her computational budget (e.g., money and time) when the sought password is long or has multiple charsets [47]. A *dictionary-based* attacker typically generates a guess list that contains diversified wordlists (e.g., common natural words, numeric patterns and first names) and candidate passwords extended through a predefined mangling ruleset [47,48]. A *probability-based* attacker attempts to describe the target password distribution by parametric probability models (e.g., Markov [37]), then search the entire key space in the descending order of probability to produce guesses. Note that real-world attackers are likely to try all these three guessing strategies to achieve higher cracking rates, and we refer to such a powerful attacker as the combined attacker.

Due to the intrinsically different characteristics of various guessing strategies, the guessabilities/strength of the same password in different scenarios will inevitably have a large gap. Even using the same training data, guessability results per password can differ by many orders of magnitude between approaches. For example, for an attacker who adopts JtR [47] with Tianya as the wordlist, she can hit the password `111222tianya` after 10 guessing attempts (see Table 10 in the full version of this paper at `https://bit.ly/3xYjsUl` for details); However, for an attacker using the Markov algorithm, even if she adopts the Tianya dataset as the training set, it will take her 83,120 attempts to hit `111222tianya`. Such a large gap in guessability estimates is due to different password guessing models used by attackers, which define which guesses should be tried and in which order [11, 39]. Therefore, the answer to **RQ2** is *negative*: There is not a single silver bullet metric to evaluate PSM in different guessing strategies.

We, *for the first time*, propose a *systematic evaluation framework* containing four different dimensioned criteria to measure PSM accuracy (see Fig. 2). Further, this framework can be extended to targeted guessing scenarios after minor modification. More specifically, in *targeted online guessing* scenarios, one can modify the ideal PSM with respect to each user (instead of the entire user group in trawling guessing scenarios); In *targeted offline guessing* scenarios, one can directly add targeted guessing strategies (e.g., [5, 40]) without proposing new metrics or altering the framework.

3.3 Metrics for online guessing

Measuring password strength hinges on determining the order in which an attacker would make guesses. Due to the limited attempts allowed in online guessing scenarios (e.g., $\leq 10^4$; see Appendix A), guessing passwords in the order of decreasing probability is the optimal strategy [16, 21]. Consequently, popular passwords are weaker because they are tried before unpopular ones. That is, the higher the frequency, the lower the strength. Based on such an optimal guessing strategy, an *ideal PSM* can be regarded as a good benchmark for evaluating PSM accuracy in online guessing scenarios, for it uses the frequency of passwords in a real-world dataset to indicate their strength [10, 16, 17]. In such a case, the accuracy of a PSM can be measured as its distance from the ideal PSM, by calculating the correlation between its evaluated strength rank list and the referred rank list of the ideal PSM. The more accurate a PSM is, the ranking of its output of password strength/probability will be closer to the ranking of the ideal PSM. Under this rationale, Kendall and Spearman correlations have long been used to rate PSM accuracy [10, 16, 17].

Classic correlation coefficients (e.g., Spearman and Kendall) are not suitable for measuring the accuracy of PSMs because they mistakenly treat all passwords as equally weighted data points [10]. Since the strength evaluation error of frequent passwords will impact more accounts, it is necessary to weight the correlation coefficients. After comparing 19 weighted metrics, Golla and Dürmuth [10] revealed that weighted Spearman correlation coefficient (WSpearman) is currently the most ideal metric to measure the accuracy of PSM for online guessing scenarios. Hence, we use WSpearman to measure PSM accuracy in online scenarios. The range of WSpearman is $[0,1]$, and the higher the value, the more accurate the PSM in online guessing scenarios.

►Weighted Spearman correlation coefficient

To the best of our knowledge, there has been no publicly-known method for calculating the WSpearman. Accordingly, we investigate six different methods and experimentally analyze their rationality for evaluating PSM accuracy (see Appendix B in the full version of this paper for details). We prefer the best one from [54], which is also adopted in [10].

We define the weighted rank vectors of the ideal PSM and the tested PSM as **X** and **Y**, respectively. The i-th ($1 \leq i \leq$n) members of **X** and **Y** are called x_i and y_i. Then WSpearman is calculated as

$$WSpearman = \frac{\sum_{i=1}^{n}\left[w_i(x_i-\bar{x})(y_i-\bar{y})\right]}{\sqrt{\sum_{i=1}^{n}\left[w_i(x_i-\bar{x})^2\right]\sum_{i=1}^{n}\left[w_i(y_i-\bar{y})^2\right]}}, \quad (1)$$

where w_i is the weight, equal to the i-th password frequency, $\bar{x}$ and $\bar{y}$ are the weighted means of **X** and **Y** respectively (see Appendix B for more details).

Note that, Golla and Dürmuth [10] mainly report discrete, numerical WSpearman values of tested passwords with two specific ranks (i.e., 1,000-th and 10,000-th of their sample set). Their approach is hard to provide the whole picture of WSpearman values of tested passwords at *every* rank. For instance, it is unknown how the WSpearman value changes as the password ranking increases. Besides, under different online guessing thresholds, the accuracy ranking of PSMs will change, so discrete Wspearman values are unlikely to provide much valuable reference for administrators to select a proper PSM on demand. Thus, we show WSpearman values of tested passwords vary with the whole rank scale in the form of a curve (rather than discrete values), see Fig. 4 here and Fig. 9 in the full version of this paper at `https://bit.ly/3xYjsUl`.

3.4 Metrics for offline guessing

As shown in Table 1, an attacker can perform large-scale (usually>10^9) guesses under offline guessing scenarios. Thus, unlike online scenarios, she does not care about whether any specific guess is hit or not, but pursues a higher cracking rate on the whole under the constraints of computing resources and time. Under this circumstance, the main focus of a PSM is to accurately distinguish passwords that are easy to be cracked. Specifically, a good PSM shall give differentiated strength ratings between the cracked and remaining (uncracked) passwords, so we use KL-divergence to quantify such differences to measure the PSM accuracy in offline scenarios.

Notably, the strength of an unpopular password cannot be estimated by its frequency [10, 17, 21]. The password probability distributions (see Fig. 3) output by fuzzyPSM [17] and PCFG-PSM [25] substantiate this: for a password with count=1, the password guessing probability ranges from 10^{-47} to 10^{-5}, meaning that an unpopular password is not necessarily a secure password. In such a circumstance, the ideal PSM is no longer a good reference in offline scenarios.

►Kullback-Leibler divergence

KL-divergence [55] (also known as relative entropy), measures how different one probability distribution is from another (benchmark) probability distribution, and its range is $[0,\infty)$. For PSM accuracy, higher KL-divergence reflects better distinguishing ability (i.e., accuracy) of a PSM. Given two discrete password strength distributions P and Q, the KL-divergence from Q to P is defined as

$$KL\,(P\parallel Q)=\sum_{i}P(i)\cdot\log\frac{P(i)}{Q(i)}. \quad (2)$$

We calculate KL-divergences of three offline guessing strategies, including brute-force, dictionary-based and probability-based guessing, covering the most representatives of main-stream guessing ideas in reality. Moreover, we propose a powerful attacker (denoted as a combined attacker) who will take full advantage of various attacks to obtain a higher coverage rate, and calculate the corresponding KL-divergence. However, this realistic attacker is out of the consideration of Galbally et al. [23]. Particularly, they ignored the more threatening probability-based guessing, which has been shown to be more effective than all attack strategies they considered (i.e., brute-force guessing, dictionary guessing and dictionary with rules guessing [11, 22]). Summing up, we consider the attacker who adopts a single attacking strategy (i.e., brute-force, dictionary-based or probability-based) and a combined strategy, and calculate the corresponding four KL-divergence respectively to evaluate PSM accuracy, which is more realistic and systematic.

►Precision and Precision^{Security}

Well-performed academic PSMs (e.g., [16–18, 25]) usually adopt fine-grained password probabilities or guess numbers as strength feedback. However, this is not friendly to users' intuition of password strength [56]. In contrast, commercial PSMs (e.g., [14, 15, 26]) generally take coarse-grained but easy-to-understand binning/scoring feedback (e.g., [Weak, Fair, Good, Strong]; see Fig. 1) [7, 10, 56, 57]. Therefore, a new question arises: How will the PSM accuracy change in response when a fine-grained feedback (of a PSM) is converted into a coarse-grained score/bin.

To address this question, we introduce the Precision metric for multi-class classification to evaluate the accuracy of a converted PSM. Specifically, an accurate scoring PSM will rate the strength of a secure password as a higher score (and that of an insecure password as a lower score). Thus, we mainly explore the most representative passwords with the lowest and highest scores. This is because the former is often rejected for being too weak, reflecting the baseline strength that the site will tolerate [2, 27, 32]; The latter one with the highest score represents the site's highest expectations for password strength. The more accurate a PSM is, the more passwords with the lowest score will be cracked, and the fewer passwords with the highest score will be cracked.

We denote the set of passwords with the lowest score as L, where the number of the cracked passwords and the remaining uncracked ones are NC_L and NR_L respectively; and the set of passwords with the highest score as H, where the number of the cracked passwords and the uncracked ones are NC_H and NR_H respectively. The weight of L is defined as

$$W_L = \frac{NR_L + NC_L}{NR_L + NC_L + NR_H + NC_H}, \tag{3}$$

and the weight of H is

$$W_H = \frac{NR_H + NC_H}{NR_L + NC_L + NR_H + NC_H}. \tag{4}$$

Thus Precision is calculated as

$$\begin{aligned} Precision &= W_L \cdot \frac{NC_L}{NR_L + NC_L} + W_H \cdot \frac{NR_H}{NR_H + NC_H} \\ &= \frac{NC_L + NR_H}{NR_L + NC_L + NR_H + NC_H}. \end{aligned} \tag{5}$$

We note that strength misjudgments in different directions (e.g., weak→strong or strong→weak) would have varied security impacts. Specifically, the strong→weak misjudgment is prone to impose a burden (frustration/fatigue) on users and thus reduce PSM usability [27]. In contrast, the weak→strong misjudgment will mislead the user that her weak password is strong enough, which has a much more negative security impact than strong→weak misjudgment. Thus, a PSM should strive to prioritize the reduction of the weak→strong misjudgments. Comparatively, the misjudgment of medium-strength passwords has a much smaller impact. To deal with this, we propose the reconciled Precision metric (i.e., Precision^{Security}) to characterize the impacts of strength misjudgments in different directions on PSM accuracy:

$$Precision^{Security} = \beta \cdot W_L \cdot \frac{NC_L}{NR_L + NC_L} + (1-\beta) \cdot W_H \cdot \frac{NR_H}{NR_H + NC_H}. \tag{6}$$

Assigning a higher weight to events with a higher risk (or more likely to occur) is a common approach in risk management and decision-making under uncertainty [58]. For linear weighting, the value 0.8 is widely preferred [59–61]. Without loss of generality, we also set the reconciled misjudgment parameter β=0.8 corresponding to the more serious misjudgment weak→strong, which indicates a four times higher significance than the strong→weak misjudgment. It is worth noting that the value of β can be adjusted according to the system's risk perceived and usability analysis, and how to optimize it is out of our scope.

4 Experimental methodology

We now show the basic information about our password datasets, introduce twelve state-of-the-art PSMs for comparison, and detail our experimental setups.

4.1 Datasets and ethical considerations

Our experiments are built on 14 widely used real-world password datasets (see details in Table 2). These datasets have different password strengths, languages and services, and have been widely used in relevant research on PSM design and evaluation (e.g., [10,13,14,17]). We select passwords with lengths no longer than 30 and only contain ASCII characters for evaluation, following the same preprocessing method in prior studies (e.g. [17, 18, 24]). This is because non-ASCII passwords and length>30 passwords are unlikely to be chosen by users, they may be junk information or randomly generated by password managers [24], while PSMs are designed to evaluate user-chosen passwords. In all, 0.00%∼2.71% (avg. 0.43%) of passwords are removed from the original datasets.

Our testing sets are quite diverse, covering services with various security requirements. For instance, CSDN requires passwords to be with length≥8, and 000webhost requires passwords with length≥6 and at least a letter and a digit (see Appendix D in the full paper for more details). Besides, since the language is an important factor that affects password strength [24, 62, 63], our work evaluates passwords from both English and Chinese sites, *filling the gap in [10, 27] that only focus on how PSMs perform on English passwords*.

Though publicly available and widely used [10, 17, 24, 37, 38], these datasets are sensitive. We store and process them on computers disconnected to the Internet, and only report the aggregated statistics to ensure that no identifiable information can be linked to the corresponding victim.

4.2 Leading PSMs to be evaluated

The effectiveness and practicality of our framework are demonstrated and examined by rating 12 leading academic and industrial PSMs. The selection of these 12 PSMs is based on the preliminary results of 45 meters (81 variants) studied in [10], 50 meters studied in [32] and 22 meters studied in [64]. *Each of them is a representative of one PSM family* in terms of the deployability, design idea, training method and time, storage space, blocklist adoption, feedback form, and quantization (see Table 3). Below are some necessary details.

PCFG-PSM/fuzzyPSM. Weir et al. [36] proposed a seminal password guessing model based on probabilistic context-free grammars (PCFG). Houshmand and Aggarwal [25] then put forward PCFG-PSM that estimates a password's probability for users to choose them. Considering users' ubiquitous password reuse behaviors, Wang et al. [17] designed fuzzyPSM on the basis of the fuzzy-PCFG algorithm. At the training stage, fuzzyPSM [17] parses each password pair in the training sets, and learns which and how mangling rules are employed by users to construct new passwords from their old ones. As the server generally does not have users' old passwords, fuzzyPSM [17] overcomes this problem by approximation: it requires a relatively weak training set and an additional stronger training set with languages, services and password policies similar to the target website.

MultiPSM. Galbally et al. [19] believed that "no password strength metric by itself is better than all other metrics for every possible password". Their MultiPSM [19] combines

Table 2: Basic information about the fourteen real-world password datasets (PWs stands for passwords).†

Language	Dataset	Web service	Leaked time	Original PWs	Miscellany	Length>30	Removed %	After cleaning	Unique PWs	Role
Chinese	Tianya	Social forum	Dec. 2011	31,761,424	860,183	5	2.71%	30,901,241	12,898,437	Training set
	Dodonew	E-commerce	Dec. 2011	16,283,140	10,774	13,475	0.15%	16,258,891	10,135,260	Test set
	Taobao	E-commerce	Jan. 2012	15,073,116	0	86	0.00%	15,073,030	11,634,170	Training set *B*
	CSDN	Programmer forum	Dec. 2011	6,428,632	0	355	0.01%	6,428,277	4,037,605	Test set
	TPYDL	IT portal	Dec. 2011	5,444,441	0	26	0.00%	5,444,415	2,884,441	Training set *B*
	Weibo	Social forum	Dec. 2011	4,730,662	0	420	0.01%	4,730,242	2,828,618	Test set
	Renren	Social forum	Dec. 2011	3,257,831	0	19	0.00%	3,257,812	1,904,776	Training set *B*
English	Rockyou	Social forum	Dec. 2009	32,603,387	18,377	3,140	0.07%	32,581,870	14,257,653	Training set
	LinkedIn	Professional social	May 2016	54,638,863	0	17,154	0.03%	54,621,709	24,681,306	Test set
	Twitter	Social forum	June 2016	40,669,963	0	282,149	0.69%	40,387,814	10,583,709	Training set *B*
	000webhost	Website service	Oct. 2015	15,299,590	0	955	0.01%	15,298,635	10,583,709	Test set
	Hostinger	Website service	May 2015	15,299,590	0	955	0.01%	15,298,635	717,641	Training set *B*
	Yahoo	Web portal	July 2012	453,491	10,657	0	2.35%	442,834	337,136	Test set
	Gmail	Email	Sep. 2014	4,926,650	0	3,120	0.06%	4,923,530	3,132,028	Training set *B*

† To characterize users' password reuse behaviors, fuzzyPSM [17] requires two training sets: a relatively weak password dataset, called training set *A* and a relatively strong training set *B*.

Table 3: Basic information about the leading PSMs for comparison (PW stands for password; Tr for training).

Source	Leading PSM	Deployed in	Design idea	Tr method	Storage (MB)†	Blocklist	Feedback form	Quantization‡
Academic	fuzzyPSM [17]	Server	Attack algorithm (fuzzy-PCFG)	Adaptive	198.66	×	PW probability	N/A
	MultiPSM [19]	Server	Multimodal	Adaptive	75.10	✓	Fusion score	N/A
	PCFG-PSM [25]	Server	Attack algorithm (PCFG)	Adaptive	108.54	×	PW probability	N/A
	Markov-PSM [16]	Server	Attack algorithm (Markov)	Adaptive	101.23	×	PW probability	N/A
	RNN-PSM [18]	Server/Client	Attack algorithm (RNN)	Adaptive	20.12	×	PW probability	N/A
	LPSE [13]	Client	Rules (Vector similarity)	Static	N/A	×	Vector similarity	Q3
	CNN-PSM [20]	Server	Probabilistic models (CNN)	Adaptive	36.00	×	PW probability	N/A
Industrial	Zxcvbn [14]	Client	Pattern detection	Static	N/A	✓	Guess number	Q5
	KeePSM [15]	Client	Pattern detection	Static	N/A	✓	Entropy	Q5
	12306-PSM	Client	Rules	N/A	N/A	×	Rating	Q3
	Microsoft-PSM [26]	Client	Rules	N/A	N/A	×	Rating	Q4

† The results shown here are measured with the canonical Rockyou dataset (see Table 2) as the training set.
‡ Q3=[Weak, Medium, Strong]; Q4=[Very weak, Weak, Strong, Perfect]; Q5=[Very weak, Weak, Medium, Strong, Perfect].

the scores provided by a blocklist, a brute-forceable password detection mechanism, and two heterogeneous Markov chains into the final fusion as the final multimodal strength feedback. In our evaluation, we adopt their executable graphical application JRC-PaStMe [65].

Markov-PSM. Castelluccia et al. [16] designed an adaptive PSM based on the Markov model. Then Ma et al. [37] improved Markov with normalization and smoothing techniques. We adopt the setting recommended by Wang et al. [24]: 4-order Markov with End-symbol Normalization and Laplace Smoothing (with δ=0.01 as suggested in [37]).

RNN-PSM. Melicher et al. [18] presented a probabilistic model based on Recurrent Neural Networks (i.e., RNN-PSM) to measure password guessability. RNN-PSM [18] predicts the next character of a password fragment and outputs the probability. It outperforms its counterparts under large guess numbers (usually$\geq 10^{10}$ [18, 22]).

LPSE. Guo and Zhang [13] employed two kinds of similarity to measure the strength of a given password, namely cosine-length similarity and password edit distance. Unlike other rule-based PSMs (e.g., 12306-PSM), LPSE [13] represents a password by a vector containing password charset and length. Then, it evaluates password strength by calculating the similarity between the two vectors of the user's password and the standard strong password (randomly generated, length$\geq$16). We choose the recommended cosine length metric of LPSE [13] for evaluation.

CNN-PSM. Pasquini et al. [20] proposed an interpretable probabilistic PSM, using a lightweight deep learning framework from Convolutional Neural Network. We call it CNN-PSM. In particular, CNN-PSM [20] disentangles the security contribution of each character in the password, and provides explicit fine-grained character-level strength feedback. But as shown in Table 3, CNN-PSM [20] requires much more (overwhelming) computing resources and training time than other PSMs (such as RNN-PSM [18]).

Zxcvbn. This PSM has been adopted by well-known services such as Dropbox and WordPress [14]. It decomposes an entered password into several patterns that may overlap, and then estimates the number of guesses required by the attacker to hit the password, and outputs the strength score. However, the original Zxcvbn [14] is primarily designed for English users (so we call it Zxcvbn$_E$), ignoring users in other languages. Thus, we replace some of its built-in English dictionaries with the corresponding Chinese dictionaries (see Appendix E in the full version of this paper for more details), and get the modified Zxcvbn$_C$ to explore **RQ4**.

KeePSM. KeePSM is the built-in PSM of the popular password manager KeePass [15]. KeePSM [15] expresses the password strength as an entropy score in bits and a colored progress bar. Similar to Zxcvbn [14], KeePSM [15] detects specific patterns of a given password. Then, it takes an optimal static entropy encoder to calculate the minimum encoding cost of pattern combinations. Compared to other PSMs, KeePSM [15] is somehow too strict. For example, a random password `0*IghdA9i?P1` containing 12 characters of four charsets is only rated as 78 bits (rate it as "weak").

12306-PSM/Microsoft-PSM. 12306.cn is the official ticketing website of China Railway, and is used by six hundred million users in their real names. 12306-PSM rates users' passwords into three levels: A "weak" password is with length$\leq$7 and composed only of letters, digits or the underscore character '_'; A "strong" password contains mixed-case letters, digits and at least one '_'; the remaining passwords are labeled as "medium". Microsoft-PSM has been adopted by popular services such as Outlook and Skype. Microsoft-PSM divides passwords into four levels: "very weak", "weak" and "strong" if the password length is < 8, within 8∼14 and >14, respectively; a "strong" password will be upgraded to "perfect" if it contains 2^{+} charsets. Such heuristic rule-based PSMs mainly consider password composition, are far from accurate and often provide misleading feedback for weak passwords [27, 64].

4.3 Experimental setup

Among the 12 PSMs to be evaluated, five of them (i.e., Markov-PSM [16], MultiPSM [19], PCFG-PSM [25], RNN-PSM [18], and CNN-PSM [20]) need one training set, and one (i.e., fuzzyPSM [17]) needs two training sets. To make our experiments as realistic as possible, our choices of the training set(s) for a given test set (simulating the target site) adhere to three rules: (1) They never come from the same service; (2) They are of the same language; and (3) The training set(s) shall be as large as possible. Rule (1) prevents our experiments from the overfitting issue, while rules (2) and (3) ensure the effectiveness of each algorithm. This gives rise to the dataset setup in Table 2. Two training sets (i.e., Tianya and Rockyou) are used by all six PSMs, and four additional datasets are used as Training set *B* (simulating password reuse behaviors) for fuzzyPSM [17]. Our training sets are also widely used in various PSM-related literature [10, 13, 14, 17].

For a fair comparison, we further make sure that all 12 PSMs work on the same test set, and manage to obtain their codes shared (implemented) by the original authors or directedly from target sites. For all parameters, we follow the best recommendations of the original authors. We adopt the entire dataset for training for the six attack algorithm-based PSMs (see Table 3). This is because, for such PSMs, their accuracy is related to the training set size. Particularly, we explored the impact of training set size on PSM accuracy (see Appendix F of the full paper for detailed experimental results). We find that compared with PSMs trained with sub-sample (i.e., smaller) datasets, the PSMs trained with the entire dataset perform better in both online and offline scenarios.

4.3.1 Online guessing scenarios

According to rule-of-thumb recommendations of [39, 40, 51] (with no concrete, real-world empirical evidence) and our manual test (see Appendix A), we choose 10^4 as the online guessing threshold. The *ideal* strength ranking of a *popular* password can be approximated as its frequency ranking of a large enough password corpus [10, 17, 21]. However, this straightforward idea is hard to apply to approximate the strength of unpopular passwords [4]. To mitigate approximation errors, we first select passwords with frequency$\geq$10 (as recommended by [10]) from each target password dataset to form the set $\mathcal{D}$. Then, we use $\mathcal{D}$ as the basis of the test set, and adopt two different strategies to select testing passwords from $\mathcal{D}$ for accuracy comparison in online scenarios:

► **For knowledgeable online attacker.** Since many sites (e.g., Yahoo [46], Twitter [66]) have leaked their user passwords more than once, a powerful (yet realistic) attacker can learn the actual password distribution of the targeted website. Due to limited guess attempts allowed, the attacker will prioritize the most popular passwords of this distribution within the throttling threshold. Thus, we take the top-10^4 popular passwords in $\mathcal{D}$ as the test set to calculate WSpearman.

► **For general online attacker.** For a general attacker unaware of the target password distribution, her sensible strategy is to traverse common popular passwords (rather than the most popular ones) of the target site. The general attacker is further confined to submitting passwords in the target password set for online guessing, while any real attacker does not have (but can only approximate) this ability. That is, our general attacker has the upper-bound capabilities of all online attackers who are unaware of the target distribution. To characterize her behavior, we *randomly* select 10^4 passwords from $\mathcal{D}$ as the *test set* and calculate the WSpearman value.

PSMs that feedback coarse-grained bins/scores (e.g., 12306 -PSM and Microsoft-PSM) often measure passwords with obviously different strengths to have the same strength level/score. However, under the experimental method in which the ideal PSM is the benchmark, such PSMs seem easier to evaluate popular passwords accurately. This is because they have orders of magnitude of fewer strength options (e.g., [weak, medium, strong]) than PSMs whose feedback is fine-grained password probabilities or guesses. To explore whether this phenomenon leads to an unfair evaluation of PSMs under the WSpearman metric, we construct an artificial *Pseudo-PSM*, which has only one strength rating, representing the extreme case of coarse-grained PSMs. That is, Pseudo-PSM rates the strength of all passwords with the same score (and thus is highly inaccurate). If its WSpearman is always very high, this indicates that our WSpearman metric cannot fairly evaluate coarse-grained PSM accuracy; If its WSpearman is always close to 0, it indicates no correlation between the password strength given by Pseudo-PSM and the ideal PSM. That is, our adopted WSpearman metric can effectively measure these coarse-grained PSMs (e.g., Microsoft-PSM [26], 12306-PSM , including apparently unreasonable Pseudo-PSM). Our experimental results well fit the latter case (see Fig. 4), indicating our adopted Wspearman has avoided giving coarse-grained PSMs an inappropriate advantage.

4.3.2 Offline guessing scenarios

We consider four most representative offline attackers (or strategies) and their configurations are as follows:

► **For brute-force offline attacker.** The guess number of the brute-force attack is related to password charset and length. We follow the method in [23] to implement 10^{12} guesses.

► **For dictionary-based offline attacker.** We select the famous JtR [47] as the dictionary-based guessing tool, adopt the training set in Table 2 as the wordlist, and conduct 10^9 guesses. As for the mangling rules, following the recommendation of Ur et al. [11], we choose Spiderlabs ruleset [67].

► **For probability-based offline attacker.** We train 4-gram Markov [37] (using end-symbol normalization and Laplace Smoothing) suggested by Wang et al. [24] on our training sets, and generate 10^9 guesses in decreasing order of probability.

► **For combined offline attacker.** This attacker is powerful enough to conduct the above three guessing attacks simultaneously to achieve a higher cracking rate, so we take the union of the above three generated sets as her guessing set.

Under each guessing strategy, we calculate the strength distributions of the cracked passwords and the remaining (uncracked) passwords, and use KL-divergence to evaluate the differences between the two distributions to measure PSM accuracy. Particularly, the higher the KL-divergence, the more accurate the PSM is in offline guessing scenarios.

To simulate the transformation from fine-grained strength feedback (e.g., entropy, guess number or probability) to easy-to-understand (widely used) bins/scores, we take the following methods to convert the output of PSMs: (1) For PSMs with their own scoring/binning strategies (i.e., LPSE [13], Zxcvbn [14], KeePSM [15], 12306-PSM and Microsoft-PSM), we directly take their scoring results; (2) Regarding probability-based fuzzyPSM [17], PCFG-PSM [25], Markov-PSM [16] and RNN-PSM [18], we first use the Monte Carlo method [35] to calculate the guess number of each testing password, and then obtain the strength scores of testing passwords using the guess number-score conversion method of Zxcvbn [14], namely: $<10^3$ (score 0), $10^3\sim10^6$ (score 1), $10^6\sim10^8$ (score 2), $10^8\sim10^{10}$ (score 3) and $>10^{10}$ (score 4); (3) MultiPSM [19] outputs a fusion score from 0 to 10 as the strength. We linearly map this score to an integer from 0 to 4 as the converted score for the sake of comparison; and (4) CNN-PSM [20] does not conform to the probability model, so the Monte Carlo method [35] is inapplicable. As a solution, we divide the password probability interval $10^{-200}\sim1$ evenly into five parts, corresponding to the strength scores 0∼4. We choose a score range 0∼4 because this is the strength scale setting for most commercial PSMs [27, 32].

To measure the accuracy of converted PSMs with the feedback of coarse-grained bins/scores, we calculate Precision and Precision^{Security} (see Sec. 3.4). The higher the metrics, the fewer misjudgments of password strength and the more accurate the PSM examined.

5 Experimental results and analysis

We now provide a comprehensive, comparative evaluation of the accuracy of 11 leading PSMs as listed in Table 3, as well as our modified Zxcvbn$_C$ (see Sec. 4.2).

5.1 Results in online guessing scenarios

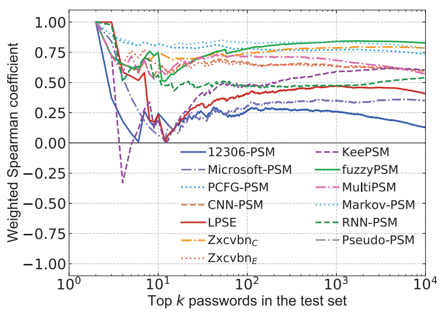
As explicated in Sec. 3, we take WSpearman as the metric to evaluate PSM accuracy under the general and knowledgeable guessing strategies in online guessing scenarios. The results on Dodonew, Weibo and LinkedIn are shown in Fig. 4. Due to space constraints, the results on CSDN, 000webhost and Yahoo are shown in Fig. 9 of the full paper.

As defined in Sec. 3.2, the knowledgeable attacker is well-informed of the target password distribution, and she will prioritize the most popular passwords (e.g., top-10^4) of this distribution [4, 21]. Almost all PSMs (excluding LPSE [13], KeePSM [15] and RNN-PSM [18]) can accurately evaluate these extremely popular passwords under the knowledgeable guessing strategy, so the WSpearman value usually starts around 1.0 (see Fig. 4). In contrast, the WSpearman value is significantly lower under the general guessing strategy where the attacker is unaware of the target password distribution.

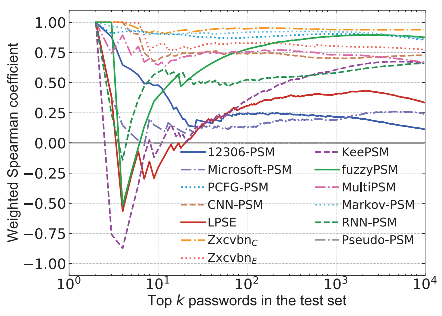
FuzzyPSM [17] always performs the best in online guessing scenarios. Under the knowledgeable guessing strategy, its WSpearman is always greater than 0.5, usually the highest. Under the general guessing strategy, it still has a significant advantage in accuracy over all other PSMs. The intrinsic reason behind its good performance is that fuzzyPSM [17] can well capture users' password reuse behavior without users' old/leaked passwords (while all other PSMs cannot).

For pattern detection-based PSMs, the WSpearman values of Zxcvbn$_C$ and Zxcvbn$_E$ [14] are significantly higher than KeePSM [15]. The reason is that Zxcvbn [14] has a comprehensive set of methods for detecting/recognizing common patterns in weak passwords, including keyboard patterns, dates and names, etc. In online guessing scenarios, Zxcvbn$_C$ performs better than Zxcvbn$_E$ [14] on Chinese datasets, while the opposite is true on English datasets. For example, Zxcvbn$_C$ measures the 18th-ranked password `woaini1314` (which is a popular password that means "I love you forever" in Chinese Pinyin) in Dodonew as *guess_number*=63. However, Zxcvbn$_E$ [14] evaluates its strength as *guess_number*=85,143,792. The significant gap between these two guess numbers suggests the necessity of adapting the dictionaries of a pattern detection-based PSM to the language of the targeted service. This suggests that the performance of pattern-based PSMs are language dependent. We further explore whether this holds on six attack algorithm-based PSMs (see Table 3), and find they all perform worse on services in languages different from which it is originally designed for (see Appendix F in the full version of this paper for details). This answers **RQ4**.

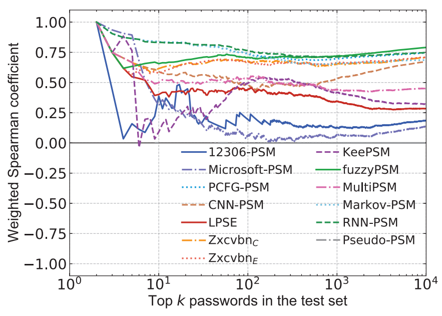
Among attack algorithm-based PSMs, fuzzyPSM [17] performs slightly better than Markov-PSM [16] and RNN-PSM [18], significantly better than PCFG-PSM [25]. Though both fuzzyPSM [17] and PCFG-PSM [25] are based on the PCFG algorithm [36], fuzzyPSM [17] captures real users' password reuse behaviors and common modification methods in its fuzzy-PCFG algorithm, which is the essential reason for its



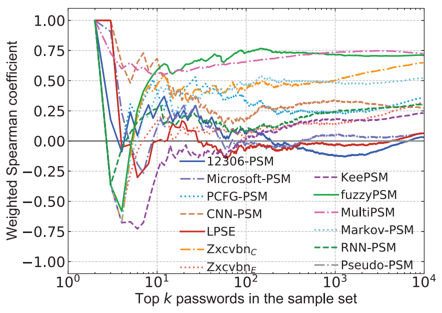
(a) Knowledgeable strategy, tr: Tianya, ts: Dodonew



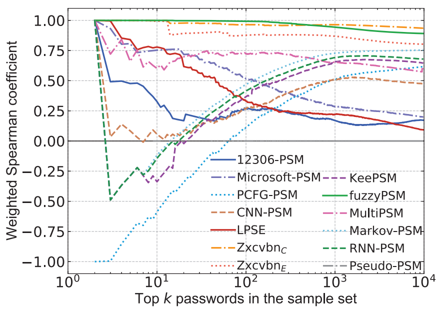
(b) Knowledgeable strategy, tr: Tianya, ts: Weibo



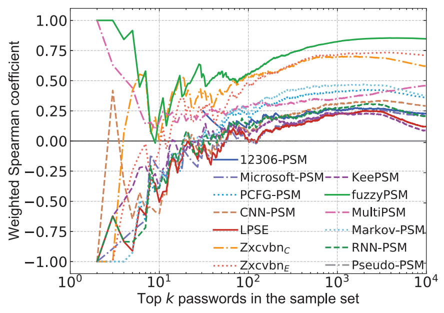
(c) Knowledgeable strategy, tr: Rockyou, ts: LinkedIn



(d) General strategy, tr: Tianya, ts: Dodonew



(e) General strategy, tr: Tianya, ts: Weibo



(f) General strategy, tr: Rockyou, ts: LinkedIn

Figure 4: Weighted spearman correlation coefficient of state-of-the-art PSMs in online guessing scenarios (tr: Training set; ts: Test set).

high accuracy. However, in most cases, attack algorithm-based PSMs (except for fuzzyPSM [17]) perform slightly worse than pattern detection-based Zxcvbn$_C$ and Zxcvbn$_E$ [14]. This is mainly because the former has a lower ability in measuring short (or simple-composed) passwords and does not deploy the blocklist or dictionary to detect common weak passwords. On the contrary, Zxcvbn$_E$ [14] and Zxcvbn$_C$ are good at characterizing such passwords due to their built-in popular-password dictionaries. Although MultiPSM [19] fuses a blocklist module and two heterogeneous Markov modules, its performance is worse than that of the single-model Markov-PSM [16] in the online guessing scenario. This implies that its heuristic fusion strength score does not fully exploit the advantages of each constituent module.

The accuracy of CNN-PSM [20] is mediocre under the knowledgeable guessing strategy, but it seems to be worse under the general guessing strategy (WSpearman<0.5). This suggests that CNN-PSM [20] can more accurately measure top-ranked passwords than measure common ones. The advanced rule-based LPSE [13] usually performs not very well, indicating that its heuristic construction of password vectors is somewhat unreasonable, or its adopted cosine-length similarity is not suitable. Rule-based 12306-PSM and Microsoft-PSM simply output the heuristic strength bin/score of a password instead of the fine-grained password probability or guess number. In online scenarios, these PSMs usually perform the worst, indicating that the crude rule-based PSMs are far from accurate. For instance, they do not rate the popular password `111222tianya` (rank=10 in Tianya dataset; see Table 10 in the full paper) as weak, just because it is composed of two charsets and its length=12. The WSpearman value of Pseudo-PSM is almost 0. This in turn demonstrates that the correction strategies in our adopted WSpearman have avoided giving coarse-grained PSMs an inappropriate advantage.

Summary. In online guessing scenarios, fuzzyPSM [17] performs the best, followed by Zxcvbn [14]. Pattern-based and attack-algorithm-based PSMs need to be adapted for evaluating passwords in different languages.

5.2 Results in offline guessing scenarios

Among our proposed three metrics for offline scenarios (see Sec. 3.4), KL-divergence can reveal the distinguishing ability of a PSM; Precision can measure PSM accuracy when its non-binned strength feedback is transformed to widely used bins/scores; and Precision^{Security} can characterize the impacts of misjudgments in different directions on PSM accuracy. The higher the metric value, the more accurate the PSM is under the corresponding offline strategy.

5.2.1 KL-divergence

In Table 4, we summarize the KL-divergence of leading PSMs under various attacking strategies, and identify the best PSM under each guessing strategy with the background color. For brute-force, MultiPSM [19] performs the best, because it contains a brute-force detection model. Rule-based LPSE [13] and Microsoft-PSM are also noteworthy: Their KL$_{Bfa}$ are the second and only lower than MultiPSM [19] in most cases. This is mainly because these rule-based PSMs evaluate password strength by password complexity (i.e., length and charsets), which just defines the password space for brute-force guessing [68]. That is, the intrinsic mechanisms of rule-based PSMs and brute-force attacks are interlinked.

Table 4: KL-divergence of leading PSMs in offline guessing scenarios. MultiPSM [19] performs best under brute-force and combined guessing strategies.†

PSM	KL	Chinese			English		
		Dodonew	CSDN	Weibo	LinkedIn	000web.	Yahoo
fuzzyPSM [17]	KL$_{Bfa}$	1.0788	2.5179	3.1533	0.8308	0.9268	0.7865
	KL$_{Dic}$	2.1009	6.7767	0.9021	3.0155	4.3028	3.7928
	KL$_{Prob}$	1.2467	1.1164	0.4478	1.3486	2.4993	1.5537
	KL$_{All}$	1.6193	3.1912	3.9749	2.2784	2.1186	3.3429
MultiPSM [19]	KL$_{Bfa}$	7.8006	2.0598	1.1105	8.5515	10.8753	6.4370
	KL$_{Dic}$	0.3519	0.2727	0.1161	0.5221	1.0597	0.2488
	KL$_{Prob}$	1.6762	1.8614	1.1436	5.4648	2.0630	2.0235
	KL$_{All}$	8.8994	7.2047	5.3692	7.5933	4.6160	3.9984
PCFG-PSM [25]	KL$_{Bfa}$	3.0046	1.4362	2.3564	2.2791	2.3951	1.2541
	KL$_{Dic}$	5.5556	5.9258	1.1966	3.4309	3.0581	1.8182
	KL$_{Prob}$	0.5047	1.1080	1.0369	1.2626	1.4452	0.9729
	KL$_{All}$	4.4277	2.3542	5.2805	6.0787	6.5533	2.9720
Markov-PSM [16]	KL$_{Bfa}$	1.2793	2.7560	3.8787	1.1933	0.9148	0.9439
	KL$_{Dic}$	3.0063	6.6061	3.8439	1.1163	1.6300	1.0145
	KL$_{Prob}$	2.0506	2.0341	0.7793	6.9454	6.9880	6.7100
	KL$_{All}$	2.2533	3.4217	3.8988	5.8993	1.3360	2.3893
RNN-PSM [18]	KL$_{Bfa}$	0.6410	0.9677	1.1538	1.0628	0.9133	0.6623
	KL$_{Dic}$	1.8715	2.9528	1.5388	3.4172	5.6578	2.0859
	KL$_{Prob}$	0.9765	1.6174	0.9335	2.6606	3.0942	1.7423
	KL$_{All}$	1.1992	1.5318	2.8269	4.1227	3.2999	6.3708
LPSE [13]	KL$_{Bfa}$	8.6483	6.7929	3.6468	4.0480	1.7946	1.3838
	KL$_{Dic}$	0.6029	0.6291	0.3974	0.3833	0.5417	0.3120
	KL$_{Prob}$	0.3492	0.4390	0.3814	1.8581	1.1519	1.2526
	KL$_{All}$	6.7505	1.8783	0.9242	6.7797	6.5693	4.6571
CNN-PSM [20]	KL$_{Bfa}$	1.2415	1.7885	1.3490	0.3299	0.7236	0.5631
	KL$_{Dic}$	4.1313	6.3760	4.1089	2.2732	2.6128	2.6266
	KL$_{Prob}$	0.3446	0.6661	0.5040	1.1542	2.5521	1.7725
	KL$_{All}$	3.0218	3.3403	2.8551	3.5520	2.1229	2.4907
Zxcvbn$_C$	KL$_{Bfa}$	2.1916	3.6398	2.9347	1.1732	1.7442	1.1001
	KL$_{Dic}$	2.6813	5.8958	2.7036	1.0440	2.1223	2.0496
	KL$_{Prob}$	0.7428	1.1549	0.7512	0.9996	1.7949	1.2733
	KL$_{All}$	2.8681	3.8673	4.4639	1.6443	2.0144	2.3132
Zxcvbn$_E$ [14]	KL$_{Bfa}$	2.8831	4.2145	2.7957	1.4231	1.5154	1.2219
	KL$_{Dic}$	4.3938	7.6722	4.0757	2.0382	3.7266	2.8444
	KL$_{Prob}$	0.9770	1.5533	1.1632	1.3931	1.8223	1.4169
	KL$_{All}$	4.3930	4.6782	5.4262	4.7708	2.6660	2.6944
KeePSM [15]	KL$_{Bfa}$	2.5697	2.2666	3.9500	2.8319	2.2526	2.8474
	KL$_{Dic}$	1.0812	2.0236	0.8438	0.1939	0.5237	0.3673
	KL$_{Prob}$	0.2886	0.4965	0.9081	1.0141	0.9805	0.9177
	KL$_{All}$	2.2322	2.6261	3.9691	1.6662	0.8765	1.3227
12306-PSM	KL$_{Bfa}$	1.6920	1.6607	0.3271	0.6190	1.6402	0.3176
	KL$_{Dic}$	0.2456	0.3219	0.1479	0.1504	0.1692	0.2607
	KL$_{Prob}$	0.0058	0.0127	0.0080	0.1873	0.4803	0.0319
	KL$_{All}$	1.4990	1.3592	0.2232	0.3698	0.8558	0.2026
Microsoft-PSM	KL$_{Bfa}$	3.7802	0.2433	0.3576	3.6622	5.6669	3.2280
	KL$_{Dic}$	0.6652	0.0574	0.1811	0.4127	0.4552	0.0112
	KL$_{Prob}$	0.0545	0.0364	0.0688	0.9957	0.8549	0.1223
	KL$_{All}$	3.3667	0.2370	0.2617	2.7974	3.0522	2.6380

†KL=KL-divergence; Bfa=Brute-force attack; Dic=Dictionary-based guessing; Prob=Probability-based guessing; All=The combined guessing. A line with background color means the corresponding PSM is the best under the given strategy.

Table 5: Precision of leading PSMs with the feedback of (transformed) bins/scores in offline guessing scenarios. LPSE [13], Zxcvbn$_E$ [14], Markov-PSM [16] and MultiPSM [19] perform the best under brute-force, dictionary-based, probability-based and combined guessing strategies, respectively.†

PSM	Prec	Chinese			English		
		Dodonew	CSDN	Weibo	LinkedIn	000web.	Yahoo
fuzzyPSM [17]	Prec$_{Bfa}$	0.6593	0.7356	0.9622	0.8099	0.9136	0.9182
	Prec$_{Dic}$	0.8320	0.8828	0.7896	0.8118	0.9722	0.9941
	Prec$_{Prob}$	0.7636	0.7042	0.1239	0.8738	0.9575	0.9750
	Prec$_{All}$	0.6842	0.7631	0.9811	0.7544	0.9604	0.9569
MultiPSM [19]	Prec$_{Bfa}$	0.8738	0.8608	0.9582	0.9676	0.6753	0.8721
	Prec$_{Dic}$	0.6134	0.6435	0.6653	0.9598	0.9358	0.9791
	Prec$_{Prob}$	0.3248	0.1462	0.1736	0.8842	0.9189	0.8433
	Prec$_{All}$	0.9971	0.9917	0.9978	0.9962	0.9918	0.9975
PCFG-PSM [25]	Prec$_{Bfa}$	0.9331	0.8492	0.8829	0.9122	0.9507	0.9057
	Prec$_{Dic}$	0.9706	0.9793	0.7531	0.9523	0.9792	0.9460
	Prec$_{Prob}$	0.3522	0.9895	0.2680	0.9886	0.9891	0.9945
	Prec$_{All}$	0.9241	0.8305	0.8136	0.8747	0.9344	0.8594
Markov-PSM [16]	Prec$_{Bfa}$	0.5706	0.5967	0.9214	0.6284	0.8954	0.7478
	Prec$_{Dic}$	0.7933	0.8058	0.8520	0.5314	0.9519	0.6039
	Prec$_{Prob}$	0.9333	0.9534	0.6133	0.9900	0.9991	0.9799
	Prec$_{All}$	0.5647	0.6038	0.9181	0.4347	0.8707	0.5056
RNN-PSM [18]	Prec$_{Bfa}$	0.6514	0.6425	0.5042	0.8778	0.9527	0.9180
	Prec$_{Dic}$	0.9535	0.9359	0.7356	0.9682	0.9961	0.9667
	Prec$_{Prob}$	0.9050	0.8540	0.7709	0.8609	0.9580	0.8537
	Prec$_{All}$	0.6270	0.6301	0.4788	0.8715	0.9503	0.9796
LPSE [13]	Prec$_{Bfa}$	1.0000	1.0000	0.5467	0.9999	0.9998	0.9992
	Prec$_{Dic}$	0.3505	0.6817	0.3272	0.5878	0.7074	0.6786
	Prec$_{Prob}$	0.2349	0.6326	0.0986	0.8030	0.7956	0.8243
	Prec$_{All}$	0.9993	0.9866	0.5465	0.9974	0.9968	0.9755
CNN-PSM [20]	Prec$_{Bfa}$	0.6975	0.7272	0.7046	0.7142	0.2628	0.5030
	Prec$_{Dic}$	0.4363	0.4505	0.4425	0.7951	0.2505	0.4415
	Prec$_{Prob}$	0.1963	0.0656	0.1274	0.6746	0.2675	0.4129
	Prec$_{All}$	0.7966	0.7869	0.7475	0.9231	0.4199	0.6666
Zxcvbn$_C$	Prec$_{Bfa}$	0.8926	0.8912	0.9586	0.8251	0.8707	0.9162
	Prec$_{Dic}$	0.8485	0.9206	0.8159	0.8404	0.9827	0.9711
	Prec$_{Prob}$	0.3210	0.2329	0.1856	0.8268	0.9728	0.9367
	Prec$_{All}$	0.9866	0.9794	0.9938	0.8831	0.9866	0.9943
Zxcvbn$_E$ [14]	Prec$_{Bfa}$	0.9561	0.9326	0.9752	0.9191	0.6990	0.9206
	Prec$_{Dic}$	0.9254	0.9325	0.8770	0.9609	0.9866	0.9796
	Prec$_{Prob}$	0.2639	0.1055	0.1942	0.9160	0.9385	0.9216
	Prec$_{All}$	0.9881	0.9802	0.9970	0.9962	0.9976	0.9986
KeePSM [15]	Prec$_{Bfa}$	0.6622	0.6939	0.9085	0.6967	0.2375	0.4374
	Prec$_{Dic}$	0.4281	0.4853	0.5685	0.7807	0.3105	0.3785
	Prec$_{Prob}$	0.2232	0.0879	0.1679	0.6598	0.2953	0.3510
	Prec$_{All}$	0.7805	0.7753	0.9678	0.9158	0.4442	0.5766
12306-PSM	Prec$_{Bfa}$	0.9105	0.8295	0.4205	0.6799	0.9406	0.9978
	Prec$_{Dic}$	0.2400	0.2859	0.2519	0.4339	0.3958	0.8217
	Prec$_{Prob}$	0.1208	0.0657	0.0655	0.4389	0.5917	0.9210
	Prec$_{All}$	0.9341	0.8643	0.4320	0.7403	0.9399	0.9995
Microsoft-PSM	Prec$_{Bfa}$	0.9979	0.9985	0.5161	0.9952	0.9822	0.9937
	Prec$_{Dic}$	0.5596	0.3134	0.3082	0.6488	0.4385	0.7556
	Prec$_{Prob}$	0.1531	0.1345	0.0837	0.7743	0.5624	0.8476
	Prec$_{All}$	0.9982	0.9929	0.5164	0.9948	0.9806	0.9937

† Prec=Precision; Other abbreviations are the same with Table 4. A line with background color means the corresponding PSM is the best under the given strategy.

Under the dictionary-based guessing strategy, Zxcvbn$_E$ [14] is quite accurate, benefiting from its well-designed pattern detection mechanism, which covers common popular passwords and various password mangling rules (e.g., leet and reverse) as well as patterns (e.g., keyboard pattern, birthday). Unlike online guessing scenarios, our modified Zxcvbn$_C$ does not show an advantage compared to Zxcvbn$_E$ [14] on Chinese datasets under dictionary-based attacks. It is believed that, Zxcvbn$_C$ will be more accurate on Chinese sites, by further modifying its dictionaries according to Chinese users' password construction habits.

In probability-based attacks, Markov-PSM [16] performs the best as expected, because such attacks use the Markov model [37]. Similarly, MultiPSM [19] contains two heterogeneous Markov modules and obtains the higher KL$_{Prob}$ on English datasets. Attack algorithm-based RNN-PSM [18] performs slightly better than PCFG-PSM [25], but both of them perform quite mediocre. This can be explained by the fact that though RNN is better than Markov in large guesses (usually $\geq 10^{10}$) [18], and PCFG is better in small guesses [24], but none is always better than Markov.

As expected, MultiPSM [19] generally performs the best under the combined guessing strategy (denoted as *All*). This is because MultiPSM [19] includes the brute-force password detection mechanism, blocklist, and two heterogeneous Markov models [37], consistent with our two (single) guessing strategies. The final multimodal score of these modules, which can reflect the password strength under the combined attacks.

Table 6: Precision^{Security} of leading PSMs with the feedback of (transformed) bins/scores in offline guessing scenarios. MultiPSM [19] is still the most accurate PSM under brute-force and combined guessing strategies.[†]

PSM	Prec^{Sec}	Chinese			English		
		Dodonew	CSDN	Weibo	LinkedIn	000web.	Yahoo
fuzzyPSM [17]	Prec^{Sec}_{Bfa}	0.2993	0.3226	0.7526	0.2772	0.2139	0.3551
	Prec^{Sec}_{Dic}	0.3228	0.3576	0.6121	0.2861	0.2586	0.3907
	Prec^{Sec}_{Prob}	0.2066	0.1582	0.0736	0.2915	0.2452	0.3767
	Prec^{Sec}_{All}	0.3249	0.3462	0.7702	0.2791	0.2571	0.3868
MultiPSM [19]	Prec^{Sec}_{Bfa}	0.7978	0.7519	0.4118	0.7809	0.7303	0.7930
	Prec^{Sec}_{Dic}	0.4884	0.5142	0.5270	0.7666	0.7235	0.7832
	Prec^{Sec}_{Prob}	0.2574	0.1163	0.1336	0.7061	0.7100	0.6745
	Prec^{Sec}_{All}	0.7883	0.7836	0.7923	0.7987	0.7729	0.7988
PCFG-PSM [25]	Prec^{Sec}_{Bfa}	0.5760	0.1749	0.6153	0.3078	0.1913	0.1878
	Prec^{Sec}_{Dic}	0.5751	0.2008	0.5245	0.3134	0.1970	0.1958
	Prec^{Sec}_{Prob}	0.0717	0.1979	0.0582	0.3183	0.1986	0.2055
	Prec^{Sec}_{All}	0.5742	0.1712	0.6014	0.3003	0.1881	0.1785
Markov-PSM [16]	Prec^{Sec}_{Bfa}	0.1732	0.2026	0.3295	0.2845	0.2009	0.3108
	Prec^{Sec}_{Dic}	0.2047	0.2407	0.2686	0.2672	0.2203	0.2922
	Prec^{Sec}_{Prob}	0.2017	0.1833	0.1732	0.3375	0.2209	0.3425
	Prec^{Sec}_{All}	0.1796	0.2156	0.3331	0.2523	0.2056	0.2764
RNN-PSM [18]	Prec^{Sec}_{Bfa}	0.1588	0.1543	0.3561	0.2476	0.1959	0.7231
	Prec^{Sec}_{Dic}	0.2174	0.2118	0.3675	0.2679	0.2051	0.7651
	Prec^{Sec}_{Prob}	0.1919	0.1964	0.1554	0.2738	0.2057	0.7682
	Prec^{Sec}_{All}	0.1543	0.1539	0.3533	0.2514	0.1960	0.7780
LPSE [13]	Prec^{Sec}_{Bfa}	0.7748	0.4265	0.4191	0.7554	0.4563	0.6329
	Prec^{Sec}_{Dic}	0.2555	0.1779	0.2437	0.4272	0.2240	0.3905
	Prec^{Sec}_{Prob}	0.1630	0.1377	0.0608	0.5980	0.2932	0.4942
	Prec^{Sec}_{All}	0.7747	0.4239	0.4191	0.7549	0.4557	0.6284
CNN-PSM [20]	Prec^{Sec}_{Bfa}	0.5580	0.5817	0.5636	0.5713	0.1951	0.4023
	Prec^{Sec}_{Dic}	0.3490	0.3604	0.3540	0.6361	0.1852	0.3532
	Prec^{Sec}_{Prob}	0.1570	0.0524	0.1019	0.5397	0.1989	0.3303
	Prec^{Sec}_{All}	0.6373	0.6295	0.5980	0.7385	0.3208	0.5333
Zxcvbn_C	Prec^{Sec}_{Bfa}	0.6983	0.6399	0.7586	0.6306	0.2743	0.6610
	Prec^{Sec}_{Dic}	0.6599	0.6580	0.6442	0.6534	0.3687	0.7073
	Prec^{Sec}_{Prob}	0.2376	0.1064	0.1398	0.6316	0.3560	0.6773
	Prec^{Sec}_{All}	0.7739	0.7107	0.7869	0.6876	0.3719	0.7260
Zxcvbn_E [14]	Prec^{Sec}_{Bfa}	0.7560	0.7253	0.7679	0.7314	0.3725	0.7027
	Prec^{Sec}_{Dic}	0.7314	0.7237	0.6890	0.7670	0.6053	0.7506
	Prec^{Sec}_{Prob}	0.2030	0.4632	0.1431	0.7290	0.5645	0.7035
	Prec^{Sec}_{All}	0.7900	0.7742	0.7867	0.7953	0.6095	0.7649
KeePSM [15]	Prec^{Sec}_{Bfa}	0.5297	0.5551	0.7268	0.5573	0.1900	0.3499
	Prec^{Sec}_{Dic}	0.3425	0.3883	0.4548	0.6245	0.2484	0.3028
	Prec^{Sec}_{Prob}	0.1785	0.0703	0.1343	0.5278	0.2362	0.2808
	Prec^{Sec}_{All}	0.6244	0.6202	0.7742	0.7326	0.3553	0.4613
12306-PSM	Prec^{Sec}_{Bfa}	0.7250	0.6603	0.3360	0.5419	0.6981	0.7981
	Prec^{Sec}_{Dic}	0.1886	0.2255	0.2011	0.3451	0.2627	0.6573
	Prec^{Sec}_{Prob}	0.0934	0.0493	0.0519	0.3492	0.4200	0.7367
	Prec^{Sec}_{All}	0.7440	0.6882	0.3451	0.5903	0.6988	0.7995
Microsoft-PSM	Prec^{Sec}_{Bfa}	0.6968	0.6880	0.7613	0.7728	0.5150	0.6976
	Prec^{Sec}_{Dic}	0.4471	0.2071	0.2455	0.5041	0.2976	0.6025
	Prec^{Sec}_{Prob}	0.1219	0.0622	0.0659	0.6044	0.3963	0.6762
	Prec^{Sec}_{All}	0.7981	0.7508	0.4121	0.7812	0.7320	0.7930

[†]Prec^{Sec}=Precision^{Security}; Other abbreviations are the same with Table 4. Background color indicates the corresponding PSM is the best under the given strategy.

Summary. In offline scenarios, MultiPSM [19] generally performs the best. Markov-PSM [16] and Zxcvbn [14] are the most accurate PSMs under probability-based and dictionary-based guessing, respectively.

5.2.2 Precision and Precision^{Security}

Commercial PSMs (e.g., 12306-PSM) usually adopt intuitive color bars with bins/scores as feedback to promote users' understanding of password strength [7, 10, 56, 57]. A natural question arises: What is the status quo when transforming the feedback of a well-performing non-binned academic PSM into an easy-to-understand (widely used) bins/scores [10]. To this end, we apply the conversion method in Sec. 4.3.2 and obtain the intuitive score distribution figures of the cracked and remaining (uncracked) passwords. Due to space constraints, here we only show the score distributions on Weibo under the combined guessing strategy in Fig. 5. For the distributions on Weibo and LinkedIn under the other typical offline guessing strategies, see Figs. 14 and 15 in the full paper[1].

As mentioned in Sec. 3.2, the focus of PSMs in offline guessing scenarios is preventing weak passwords from registering successfully as much as possible, reducing the number of passwords an attacker can successfully crack with limited computing resources. In this direction, an accurate scoring PSM shall follow: A higher score corresponds to a smaller proportion of the cracked passwords and a larger proportion of the remaining passwords. In a score distribution figure, the upper and lower parts of the X-axis are score distributions of the cracked and remaining (un-cracked) passwords. Under this criterion, Markov-PSM [16] and RNN-PSM [18] perform better. Specifically, KeePSM [15] has a particularly high proportion of passwords with score=0 (i.e., very weak), indicating that its strength evaluation method is overly strict. CNN-PSM [20] is also slightly strict, which reminds us that our score conversion method (i.e., linear mapping) for CNN-PSM [20] may need to be adjusted.

Further, to precisely quantify PSM accuracy when its fine-grained strength feedback is transformed to a coarse-grained score, we adopt our proposed Precision and show the results in Table 5. We also calculate our designed reconciled Precision^{Security} value for each PSM, considering the security impact of strength misjudgments in different directions on PSM accuracy, and show the results in Table 6.

Regarding Precision, LPSE [13], Zxcvbn_E [14] and Markov-PSM [16] perform the best under brute-force, dictionary-based and probability-based attacks, respectively. It is surprising because the non-binned LPSE [13] has a relatively low KL-divergence under all four offline guessing strategies. But the binned-LPSE's [13] Precision is markedly higher than its counterparts under brute-force attacks, when its feedback is converted to scores. One possible reason is that scoring can, to some extent, mitigate strength misjudgments of a fine-grained PSM, if the score conversion method is suitable. We confirmed this conjecture through experiments and see Sec. 6 for more details.

Regarding Precision^{Security}, MultiPSM [19] is the most accurate PSM under brute-force guessing, and Zxcvbn_E shows its advantage under dictionary-based and probability-based guessing strategies. Especially, MultiPSM [19] still has the highest Precision and Precision^{Security} under the combined attacks, indicating that our adopted score conversion method has good robustness in offline guessing scenarios.

6 Insights and suggestions

In this section, we share some insights obtained from our experiments, and provide a few workable suggestions to help

[1]In Figs. 5, 14 and 15, some seemingly empty bins correspond to tiny proportions of passwords and cannot be noticeably observed.

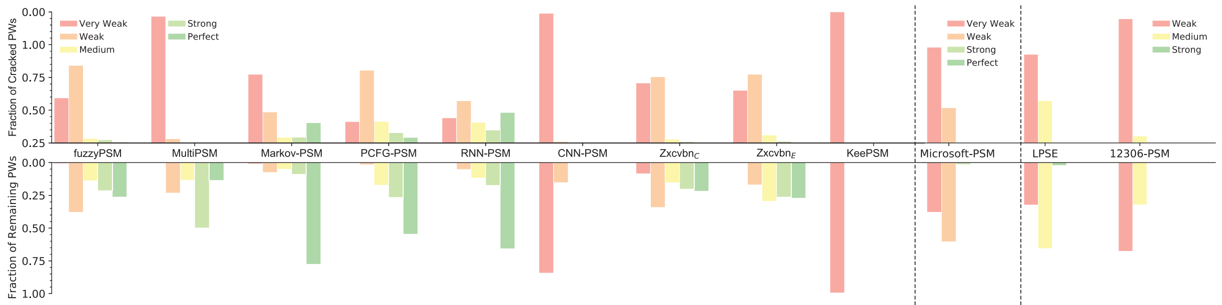


Figure 5: Password score distributions (on Weibo) of leading PSMs under the combined guessing strategy. The larger fractions of cracked low-score and remaining high-score passwords indicate a more accurate PSM. For more results on Weibo and LinkedIn, see Figs. 14 and 15 in the full version of this paper.

design a more accurate, practical, and reliable PSM.

► **Adaptive score conversion methods can be used to facilitate PSM application.**

Commercial PSMs often quantify password strength with coarse-grained 3∼5 bins/scores (see Fig. 1) [27, 32, 64]. Even in the absence of guidance, such PSMs can still provide users with a concrete, intuitive indication of when they are doing well or bad [6, 69]. Academic PSMs (e.g., fuzzyPSM [17]) are more accurate than commercial PSMs, and often take fine-grained probability, guess number, or entropy as strength feedback. Such probability or guess number can finely reflect how subtle changes in a password affect its strength [10, 20, 27]. However, due to the lack of the "existing motivator" (whether the password is "weak", "medium," or "strong") [8], it is difficult for users to understand such abstract feedback intuitively. Therefore, an adaptive score conversion method (i.e., fine-grained feedback→coarse-grained feedback) is helpful for the vast majority of non-expert users, and is expected to facilitate the adoption of more accurate academic PSMs.

When attempting to convert the fine-grained strength feedback of academic PSMs into understandable (but coarse-grained) scores, a key concern is whether this conversion will lead to accuracy reduction. Our evaluation results in Sec. 5.2 show that the non-binned LPSE [13] performs poorly (with lower KL-divergence), but surprisingly, its binned version has higher Precision and Precision^{Security} than its counterparts under brute-force guessing attacks. This implies that a score conversion method can partially mitigate the *inaccuracies* of some fine-grained PSMs in offline guessing. Furthermore, we confirm this conjecture by comparing the KL-divergence of fine-grained PSMs and their binned versions (see Sec. 4.3.2 for conversion methods), and the results are summarized in Table 12 of the full apaper. Table 12 shows that the performance of binned fuzzyPSM [17] and binned LPSE [13] increases over corresponding fine-grained versions under the combined strategy, when they evaluate English passwords.

We now further explore whether score conversion still works in online guessing scenarios. Since our results on different test sets are similar/robust, here we only show the result on the knowledgeable guessing strategy using the LinkedIn dataset in Fig. 6 (see more results in Fig. 10 of the full paper). By comparing Fig. 6 with Fig. 4(c), one can find that, overall,

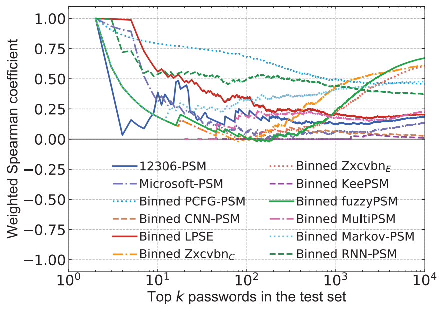


Figure 6: WSpearman correlation coefficient of 12 binned-PSMs under the knowledgeable online guessing strategy (tr: Rockyou, ts: LinkedIn). See more results in Fig. 10 of the full paper at `https://bit.ly/3xYjsUl`.

the binned RNN-PSM [18] performs better than their corresponding non-binned versions, while binned fuzzyPSM [17] and Zxcvbn [14] degrades. Our results call for effective, adaptive score conversion methods to balance PSMs' accuracy and feedback granularity, and to facilitate their application. In particular, our proposed Precision and Precision^{Security} can well quantify the accuracy of converted PSMs. In summary, we answer **RQ5** with concrete evidence that proper adaptive score conversion methods can, to some extent, mitigate inaccurate strength evaluation of a fine-grained PSM by quantization, and may facilitate the popularity of academic PSMs.

► **PSMs can be effectively integrated to perform better.**

Essentially, password strength should indicate the effort the attacker has to pay to guess the password. However, with the continuous advancement of computing technologies and the increasing diversity of guessing scenarios as well as guessing strategies, attackers often adopt various guessing methods [19, 49]. Therefore, the estimation of guessing effort (i.e., password strength) varies greatly depending on password guessing scenarios (and strategies) considered [18, 38, 64].

A desirable PSM should always provide accurate strength feedback in every guessing scenario and strategy. However, PSM designers usually only evaluate their proposed PSM under a specific guessing scenario (or strategy) that they primarily concern, and claim that this PSM is superior to other PSMs (e.g., [13, 17, 19]). Such unsystematic and unfair evaluations would confuse web administrators. According to our results in Sec. 5, none of the existing leading PSMs is exclusively better than the others. For example, fuzzyPSM [17]

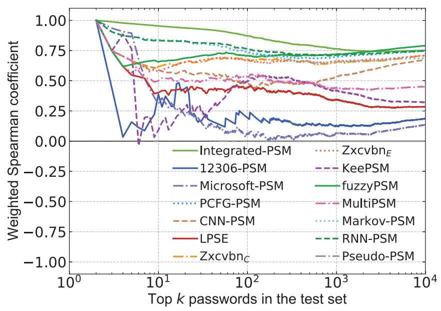


Figure 7: Weighted spearman correlation coefficient of Integrated-PSM and other 12 PSMs under the knowledgeable online guessing strategy (tr: Rockyou, ts: LinkedIn). See Fig. 11 of the full paper for more results.

performs the best in online guessing, while Zxcvbn [14] is the most accurate PSM under dictionary-based offline guessing.

By combining the advantages of different strength evaluation methods, it is possible to design a hybrid strength meter that can overcome the shortcomings of individual PSMs. Galbally et al. [19] took the first step. Their proposed MultiPSM [19, 23] has two heterogeneous Markov modules. Besides, the built-in blocklist and brute-force detection mechanisms of MultiPSM [19, 23] can identify many popular passwords. Unfortunately, we find that there is still a considerable gap between MultiPSM [19] and the most accurate fuzzyPSM [17] in online guessing scenarios (see Sec. 5.1).

Our evaluation results in Sec. 5 show that fuzzyPSM [17] performs the best in online guessing scenarios, for it adopts the sophisticated password reuse detection algorithm (i.e., fuzzy-PCFG) to more accurately capture users' "clever" modification behaviors [5, 14, 17], while the blocklist and dictionary detection mechanisms in MultiPSM cannot. Therefore, properly integrating the online-best fuzzyPSM [17] into the offline-best MultiPSM [19] will hopefully significantly improve MultiPSM's accuracy in online guessing scenarios.

Through a preliminary exploration, we show the feasibility of the above conjecture. To construct an Integrated-PSM, we perform a simple yet effective operation on the outputs of MultiPSM [19] (denoted as $Strength_{\text{Multi}}(\cdot)$) and fuzzyPSM [17] (denoted as $Strength_{\text{fuzzy}}(\cdot)$). More specifically, the password strength output by the Integrated-PSM is calculated as $Strength_{\text{Integ}}(\cdot) = Strength_{\text{Multi}}(\cdot) - \lg\left(Strength_{\text{fuzzy}}(\cdot)\right)$. The higher the $Strength_{\text{Integ}}(\cdot)$, the higher the password strength. The intuitions underlying this heuristic formula to calculate $Strength_{\text{Integ}}(\cdot)$ are that: (1) FuzzyPSM [17] outputs the password probability $p \in$[0, 1] as its feedback, while MultiPSM [19] takes the multimodal score $s \in$[0, 10] as its feedback and the strength increasing trend of its output is the opposite of fuzzyPSM [17]; (2) To make $Strength_{\text{fuzzy}}(\cdot)$=$p \in$[0, 1] have a similar scale (i.e., no order of magnitude difference) *and* the same monotonicity with $Strength_{\text{Multi}}(\cdot)$=$s \in$[1, 10], a viable solution is to take $-\lg\left(Strength_{\text{fuzzy}}(\cdot)\right)$ as the output of the fuzzyPSM module of the Integrated-PSM.

We have evaluated this Integrated-PSM in the online guessing scenario (see Fig. 7; more results are shown in Fig. 11 of the full version of this paper). Though simple, this Integrated-PSM does significantly outperform all examined PSMs in online guessing scenarios, including fuzzyPSM [17] and MultiPSM [19], especially in online knowledgeable strategy. We further evaluate its effectiveness in the offline guessing scenario by comparing its KL-divergence with other examined PSMs (see Table 13 in the full paper). Table 13 shows that, in most cases, Integrated-PSM has a higher KL-divergence under brute-force, probability-based and combined attacks compared with its components, i.e., fuzzyPSM [17] and MultiPSM [19]. This implies that our integration method will not lead to a significant loss of accuracy of PSM in offline guessing scenarios. Summing up, integrating the advantages of different PSMs is a practical way to improve the accuracy of password strength evaluation.

► PSMs need to be modified and adapted to accommodate different languages.

It has been well known that English and Chinese passwords have significant differences in characteristics [24, 37, 62, 63]. In this regard, it is unsuitable to apply the existing PSMs originally designed for English-speaking users (e.g., Zxcvbn [14]) to passwords chosen by non-English speakers. Fortunately, pattern detection-based PSM (e.g., Zxcvbn [14]) can be simply tuned to be more effective, since they do not require a time-consuming and computationally expensive training process. Within limited modification cost, we replace specific dictionaries (i.e., common passwords, surnames and first names) in Zxcvbn [14] with appropriate Chinese dictionaries to construct Zxcvbn$_C$ (see Appendix E in the full version of this paper). Compared with the original Zxcvbn [14], Zxcvbn$_C$ can more accurately evaluate Chinese password strength under online guessing scenarios. To further improve Zxcvbn [14], especially in offline guessing scenarios, we recommend adding appropriate dictionaries (such as common Chinese words). These recommendations can also be applied to modify other pattern-based PSMs to accommodate different languages.

By exploiting the knowledge of the target user's linguistic background, attackers can speed up password guessing and achieve higher cracking rates [40, 63, 70, 71]. Wang et al. [24] have confirmed this and recommended that password guessing algorithms should be trained on datasets similar to the target password, by considering the confounding factors in the order of language, service, and password policies. Naturally, this principle can be applied to configure or train PSMs based on attack algorithms (such as fuzzyPSM [17]) to ensure accuracy. To support this, we conduct a cross-language evaluation to explore the performance of six attack algorithm-based PSMs. Specifically, we use Tianya-trained PSMs to evaluate passwords from English speakers, and Rockyou-trained PSMs to evaluate passwords from Chinese speakers, then calculate the corresponding Wspearman values and KL-divergence (see Appendix F of the full paper for detailed results).

By comparing Fig. 4 here, Figs. 9 and 12 of the full paper, we can see that the change of training sets' language reduces the accuracy of six tested PSMs to different degrees in online scenarios. In the offline guessing scenario, the language change in the training set reduces the overall KL-divergence value of all tested PSMs, indicating that these PSMs cannot accurately distinguish between cracked and uncracked passwords (see Table 14 of the full paper). All this suggests that well-designed PSMs need to be configured and adapted to accommodate different languages.

7 Conclusion

In this paper, we have provided a negative answer to the question of whether there is a single metric that can thoroughly (and fairly) measure the accuracy of a password strength meter (PSM). Accordingly, we, for the first time, proposed a systematic evaluation framework composed of four different dimensioned criteria to measure PSM accuracy in various guessing scenarios and strategies. Considering the ubiquitous usage and crucial role of PSMs, we believe that a thorough and fair evaluation of the accuracy of PSMs is of practical significance, and our work takes a substantial step forward and will trigger research interest in PSM design and evaluation.

Acknowledgment

The authors are grateful to the shepherd and anonymous reviewers for their invaluable comments. Ding Wang is the corresponding author. This research was in part supported by the National Natural Science Foundation of China under Grants Nos. 62222208 and 62172240, and Natural Science Foundation of Tianjin, China under Grant No. 21JCZDJC00190. See the full version of this paper at `https://bit.ly/3xYjsUl`.

References

[1] M. Golla, M. Wei, J. Hainline, L. Filipe, M. Dürmuth, E. M. Redmiles, and B. Ur, "“what was that site doing with my facebook password?”: Designing password-reuse notifications," in *Proc. ACM CCS 2018*.

[2] K. Lee, S. Sjöberg, and A. Narayanan, "Password policies of most top websites fail to follow best practices," in *Proc. SOUPS 2022*.

[3] Z. Huang, E. Ayday, J. Hubaux, and A. Juels, "GenoGuard: Protecting genomic data against brute-force attacks," in *Proc. IEEE S&P 2015*.

[4] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Trans. Inform. Foren. Secur.*, vol. 12, no. 11, pp. 2776–2791, 2017.

[5] B. Pal, T. Daniel, R. Chatterjee, and T. Ristenpart, "Beyond credential stuffing: Password similarity models using neural networks," in *Proc. IEEE S&P 2019*, pp. 417–434.

[6] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. L. Mazurek, T. P. admin Richard Shay, T. Vidas, L. Bauer, N. Christin, and L. F. Cranor, "How does your password measure up? The effect of strength meters on password creation," in *Proc. USENIX SEC 2012*, pp. 65–80.

[7] J. Tan, L. Bauer, N. Christin, and L. F. Cranor, "Practical recommendations for stronger, more usable passwords combining minimum-strength, minimum-length, and blocklist requirements," in *Proc. ACM CCS 2020*, pp. 1407–1426.

[8] S. Egelman, A. Sotirakopoulos, I. Muslukhov, K. Beznosov, and C. Herley, "Does my password go up to eleven?: The impact of password meters on password selection," in *Proc. ACM CHI 2013*.

[9] S. Van Acker, D. Hausknecht, W. Joosen, and A. Sabelfeld, "Password meters and generators on the web: From large-scale empirical study to getting it right," in *Proc. CODASPY 2015*, pp. 253–262.

[10] M. Golla and M. Dürmuth, "On the accuracy of password strength meters," in *Proc. ACM CCS 2018*, pp. 1567–1582.

[11] B. Ur, S. M. Segreti, L. Bauer, N. Christin, L. F. Cranor, S. Komanduri, D. Kurilova, M. L. Mazurek, W. Melicher, and R. Shay, "Measuring real-world accuracies and biases in modeling password guessability," in *Proc. USENIX SEC 2015*, pp. 463–481.

[12] W. Burr, D. Dodson, R. Perlner, and et al., "NIST SP800-63: Electronic authentication guideline," NIST, Reston, VA, Tech. Rep., April 2006.

[13] Y. Guo and Z. Zhang, "LPSE: Lightweight password-strength estimation for password meters," *Comput. Secur.*, vol. 73, pp. 507–518, 2018.

[14] D. L. Wheeler, "zxcvbn: Low-budget password strength estimation," in *Proc. USENIX SEC 2016*, pp. 157–173.

[15] D. Reichl, *KeePass, version 1.26/2.23: Password Quality Estimation*, July 2015, http://keepass.info/help/kb/pw_quality_est.html.

[16] C. Castelluccia, M. Dürmuth, and D. Perito, "Adaptive password-strength meters from markov models," in *Proc. NDSS 2012*, pp. 1–14.

[17] D. Wang, D. He, H. Cheng, and P. Wang, "fuzzyPSM: A new password strength meter using fuzzy probabilistic context-free grammars," in *Proc. IEEE/IFIP DSN 2016*, pp. 595–606.

[18] W. Melicher, B. Ur, S. M. Segreti, S. Komanduri, L. Bauer, N. Christin, and L. F. Cranor, "Fast, lean, and accurate: Modeling password guessability using neural networks," in *Proc. USENIX SEC 2016*, pp. 1–17.

[19] J. Galbally, I. Coisel, and I. Sanchez, "A new multimodal approach for password strength estimation - Part I: Theory and algorithms," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 12, pp. 2829–2844, 2017.

[20] D. Pasquini, G. Ateniese, and M. Bernaschi, "Interpretable probabilistic password strength meters via deep learning," in *Proc. ESORICS 2020*.

[21] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *Proc. IEEE S&P 2012*, pp. 538–552.

[22] Q. Dong, C. Jia, F. Duan, and D. Wang, "RLS-PSM: A robust and accurate password strength meter based on reuse, leet and separation," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4988–5002, 2021.

[23] J. Galbally, I. Coisel, and I. Sanchez, "A new multimodal approach for password strength estimation - Part II: Experimental evaluation," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 12, pp. 2845–2860, 2017.

[24] D. Wang, P. Wang, D. He, and Y. Tian, "Birthday, name and bifacial-security: Understanding passwords of Chinese web users," in *Proc. USENIX SEC 2019*, pp. 1537–1555.

[25] S. Houshmand and S. Aggarwal, "Building better passwords using probabilistic techniques," in *Proc. ACSAC 2012*, pp. 109–118.

[26] *Microsoft registration*, https://account.microsoft.com/.

[27] X. de Carné de Carnavalet and M. Mannan, "From very weak to very strong: Analyzing password-strength meters," in *Proc. NDSS 2014*.

[28] F. Bergadano, B. Crispo, and G. Ruffo, "High dictionary compression for proactive password checking," *ACM Trans. Inf. Syst. Secur.*, vol. 1, no. 1, pp. 3–25, 1998.

[29] M. Bishop and D. V. Klein, "Improving system security via proactive password checking," *Comput. Secur.*, vol. 14, no. 3, pp. 233–249, 1995.

[30] J. J. Yan, "A note on proactive password checking," in *Proc. ACM NSPW 2001*, pp. 127–135.

[31] P. A. Grassia, E. M. Newton, and R. A. P. et al., "NIST SP 800-63B Digital identity guidelines: Authentication and lifecycle management," June 2017, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf.

[32] D. Wang and P. Wang, "The emperor's new password creation policies: An evaluation of leading web services and the effect of role in resisting against online guessing," in *Proc. ESORICS 2015*, pp. 456–477.

[33] M. Weir, S. Aggarwal, M. P. Collins, and H. Stern, "Testing metrics for password creation policies by attacking large sets of revealed passwords," in *Proc. ACM CCS 2010*, pp. 162–175.

[34] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. López, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in *Proc. IEEE S&P 2012*, pp. 523–537.

[35] M. Dell'Amico and M. Filippone, "Monte carlo strength evaluation: Fast and reliable password checking," in *Proc. ACM CCS 2015*.

[36] M. Weir, S. Aggarwal, B. De Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *Proc. IEEE S&P 2009*, pp. 391–405.

[37] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in *Proc. IEEE S&P 2014*, pp. 689–704.

[38] D. Pasquini, M. Cianfriglia, G. Ateniese, and M. Bernaschi, "Reducing bias in modeling real-world password strength via deep learning and dynamic dictionaries," in *Proc. USENIX SEC 2021*, pp. 821–838.

[39] D. Florêncio, C. Herley, and P. C. van Oorschot, "An administrator's guide to internet password research," in *Proc. USENIX LISA 2014*.

[40] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *Proc. ACM CCS 2016*, pp. 1242–1254.

[41] S. Jarecki, H. Krawczyk, and J. Xu, "OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks," in *Proc. EUROCRYPT 2018*, vol. 10822, pp. 456–486.

[42] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled web of password reuse," in *Proc. NDSS 2014*, pp. 1–15.

[43] R. Veras, C. Collins, and J. Thorpe, "A large-scale analysis of the semantic password model and linguistic patterns in passwords," *ACM Trans. Priv. Secur.*, vol. 24, no. 3, pp. 1–21, 2021.

[44] D. Freeman, S. Jain, M. Dürmuth, B. Biggio, and G. Giacinto, "Who are you? A statistical approach to measuring user authenticity," in *Proc. NDSS 2016*, pp. 1–15.

[45] D. Florêncio, C. Herley, and P. C. van Oorschot, "Pushing on string: The "don't care" region of password strength," *Commun. ACM*, vol. 59, no. 11, pp. 1–9, 2016.

[46] S. Larson, *Yahoo's 2013 Email Hack Actually Compromised Three Billion Accounts*, Oct. 2017, https://money.cnn.com/2017/10/03/technology/business/yahoo-breach-3-billion-accounts/index.html.

[47] P. Alexander, *John the Ripper community build (1.9.0-bleeding jumbo)*, 2019, http://www.openwall.com/john/.

[48] S. J, *Hashcat*, 2018, https://hashcat.net/hashcat/.

[49] E. Liu, A. Nakanishi, M. Golla, D. Cash, and B. Ur, "Reasoning analytically about password-cracking software," in *IEEE S&P 2019*.

[50] S. Wiefling, M. Dürmuth, and L. L. Iacono, "More than just good passwords? A study on usability and security perceptions of risk-based authentication," in *Proc. ACSAC 2020*, pp. 203–218.

[51] D. Wang, Y. Zou, Q. Dong, Y. Song, and X. Huang, "How to attack and generate honeywords," in *Proc. IEEE S&P 2022*, pp. 966–983.

[52] D. Goodin, "Anatomy of a hack: How crackers ransack passwords like "qeadzcwrsfxv1331"," May 2013, https://arstechnica.com/information-technology/2013/05/how-crackers-make-minced-meat-out-of-your-passwords/.

[53] M. Hachman, "How to create strong, secure passwords by learning how to crack them," May 2021, https://www.pcworld.com/article/394029/how-to-create-strong-secure-passwords.html.

[54] *wCorr: Weighted Correlations*, 2021, https://cran.r-project.org/web/packages/wCorr/index.html.

[55] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. of Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[56] M. Ciampa, "A comparison of password feedback mechanisms and their impact on password entropy," *Inf. Manag. Comput. Secur.*, vol. 21, no. 5, pp. 344–359, 2013.

[57] A. Vance, D. Eargle, K. Ouimet, and D. W. Straub, "Enhancing password security through interactive fear appeals: A web-based field experiment," in *Proc. HICSS 2013*, pp. 2988–2997.

[58] B. Ezell, C. J. Lynch, and P. T. Hester, "Methods for weighting decisions to assist modelers and decision analysts: A review of ratio assignment and approximate techniques," *Applied Sciences*, vol. 11, no. 21, p. 10397, 2021.

[59] W. H. Hsu, L. S. Kennedy, and S.-F. Chang, "Video search reranking through random walk over document-level context graph," in *Proc. ACM MM 2007*, pp. 971–980.

[60] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proc. EP 1998*, pp. 591–600.

[61] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu, "Sketch2photo: Internet image montage," *ACM Trans. Graphics*, vol. 28, no. 5, pp. 1–10, 2009.

[62] Z. Li, W. Han, and W. Xu, "A large-scale empirical analysis of chinese web passwords," in *Proc. USENIX SEC 2014*, pp. 559–574.

[63] S. Ji, S. Yang, X. Hu, W. Han, Z. Li, and R. A. Beyah, "Zero-sum password cracking game: A large-scale empirical study on the crackability, correlation, and security of passwords," *IEEE Trans. Dependable Secur. Comput.*, vol. 14, no. 5, pp. 550–564, 2017.

[64] X. de Carné de Carnavalet and M. Mannan, "A large-scale evaluation of high-impact password strength meters," *ACM Trans. Inform. Syst. Secur.*, vol. 18, no. 1, 2015.

[65] *Password strength meter*, 2019, https://github.com/ec-jrc/jrcpastme.

[66] M. X. Heiligenstein, *Twitter Data Breaches: Full Timeline Through 2023*, Jan. 2023, https://firewalltimes.com/twitter-data-breach-timeline/.

[67] *SpiderLabs/KoreLogic-Rules*, 2012, https://github.com/SpiderLabs/KoreLogic-Rules.

[68] C. Neskey, *Are Your Passwords in the Green?*, https://www.hivesystems.io/blog/are-your-passwords-in-the-green.

[69] S. Furnell, "Assessing password guidance and enforcement on leading websites," *Comput. Fraud Secur.*, vol. 2011, no. 12, pp. 10–18, 2011.

[70] M. Xu, C. Wang, J. Yu, J. Zhang, K. Zhang, and W. Han, "Chunk-level password guessing: Towards modeling refined password composition representations," in *Proc. ACM CCS 2020*, pp. 5–20.

[71] D. Pasquini, A. Gangwal, G. Ateniese, M. Bernaschi, and M. Conti, "Improving password guessing via representation learning," in *Proc. IEEE S&P 2021*, pp. 1382–1399.

[72] J. Blocki and W. Zhang, "DALock: Password distribution-aware throttling."

[73] B. Lu, X. Zhang, Z. Ling, Y. Zhang, and Z. Lin, "A measurement study of authentication rate-limiting mechanisms of modern websites," in *Proc. ACSAC 2018*, pp. 89–100.

[74] P. Wang, H. Gao, Q. Rao, S. Luo, Z. Yuan, and Z. Shi, "A security analysis of CAPTCHAs with large character sets," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 6, pp. 2953–2968, 2021.

[75] Y. Gao, H. Gao, S. Luo, Y. Zi, S. Zhang, W. Mao, P. Wang, Y. Shen, and J. Yan, "Research on the security of visual reasoning CAPTCHA," in *Proc. USENIX 2021*, pp. 3291–3308.

[76] M. I. Hossen, Y. Tu, M. F. Rabby, M. N. Islam, H. Cao, and X. S. Hei, "An object detection based solver for Google's image reCAPTCHA v2," in *Proc. RAID 2020*, pp. 269–284.

[77] D. Wang, H. Cheng, P. Wang, J. Yan, and X. Huang, "A security analysis of honeywords," in *Proc. NDSS 2018*, pp. 1–15.

A Online guessing threshold

To resist online password guessing, services are recommended to deploy rate-limiting mechanisms, such as account lockout and login throttling [31]. Account lockout policy aims at resisting potential online guessing attacks, by disabling the account for a preset period of time [50]. Login throttling aims to prevent high-rate login attempts, by using CAPTCHA, mobile phone verification code, etc [72]. However, the adversary could bypass lockout by changing the IP address or clearing the browser cookies [73]. Worse, most CAPTCHA systems can be easily solved by automated tools [74, 75], so the rate-limiting mechanism fails to play its expected role in the real world. As revealed by Lu et al. [73], 131 out of 182 (72%) investigated high-profile sites "allow frequent, unsuccessful login attempts without account lockout or login throttling" and for these sites, attackers can achieve an attack rate higher than 85 login attempts per day (significantly more than the NIST recommended threshold of 100 attempts for 30 days).

Table 7: Rate-limit mechanisms of Alexa top-10 websites and estimated number of login attempts per day and month.†

Alexa rank	Website	Rate-limiting mechanisms	Attempts per day	Attempts per month
1	Google	After 15 consecutive fail logins, solving CAPTCHA for subsequent login is required.	1,440	43,200
2	Baidu	After 10 consecutive fail logins, solving CAPTCHA for subsequent login is required.	1,440	43,200
3	Facebook	Allow only 20 consecutive failed logins per hour.	480	14,400
4	Bilibili	Allow only 10 consecutive failed logins per hour.	240	7,200
5	Reddit	Allow only 10 consecutive failed logins in 10 minutes.	720	21,600
6	QQ	Solve CAPTCHA for each login.	1,440	43,200
7	Bing	Allow only 10 consecutive failed logins per hour.	240	7,200
8	Yahoo	Solve re-CAPTCHA for each login.	1,440	43,200
9	Twitter	Allow only 15 consecutive failed logins per hour.	360	10,800
10	Amazon	Allow only 5 consecutive failed logins per hour.	120	3,600

† We set the interval between consecutive guesses as 1 min, and use this interval to estimate the number of login attempts allowed per day and month. In reality, various automated tools can help attackers shorten the interval (even to 19.93s [76]), so an attacker can submit more guessing attempts (compared with the number listed in this table) per day and month.

Since the authentication system has to balance online guessing attacks and denial-of-service (DoS) attacks, the failed login threshold $\mathcal{T}$ should not be too small or too large [39]. Without loss of generality, we set $\mathcal{T} = 10^4$ according to [33, 39, 42], and this threshold $\mathcal{T} = 10^4$ has been widely preferred, see [5, 51, 77]. To further support our threshold selection, we manually investigate the rate-limit mechanisms of sites[2] with Alexa rank top-10, collect the rate-limit mechanism of each site and examine the maximum number of online guessing attempts allowed (see Table 7). During the testing process, we strictly followed ethical guidelines: We use accounts created by ourselves (no other users' accounts are involved) for testing, and all login attempts are manually submitted to the website to avoid overwhelming normal requests and performance degradation to the web servers.

According to Table 7, we find that all of the investigated sites allow more than 10^3 guessing attempts in a month. Disturbingly, even these high-profile sites do not properly implement the rate-limiting mechanisms, and are vulnerable to diverse online guessing attacks. Among these ten websites, Amazon adopts the strictest lockout policy, allowing only five failed login attempts in one hour. Even so, an attacker can still submit 3,600 guessing attempts in a month. Without loss of generality, we set the online guessing threshold $\mathcal{T} = 10^4$.

[2] https://www.expireddomains.net/alexa-top-websites/, accessed on Aug. 20, 2022.

B Formulas of our adopted weighted Spearman correlation coefficient

The weighted Spearman correlation coefficient (WSpearman) refers to the weighted Pearson coefficient of two weighted rank vectors. Generally, for an unweighted rank vector, the highest rank value is 1, the second-highest rank value is 2, and so on until the nth value. The rank values of elements with the same value should be the same. A feasible method is to use the mean of all tied ranks. For n tied elements with the same value, the vector of the tied ranks is $\mathbf{v} = [a_j+1, a_j+2, \ldots, a_j+n]^T$, and their ranks are the same, which are

$$rank_j = \frac{1}{n}\sum_{i=1}^{n}(a_j+i) = \frac{1}{n}[na_j + \frac{n(n+1)}{2}] = a_j + \frac{n+1}{2}. \quad (7)$$

However, for a weighted rank vector, the weighted rank of a single element is calculated by two items. For the jth element, the rank is

$$rank_j = a_j + b_j. \quad (8)$$

The first item a_j is the sum of the weights of all elements which are less than or equal to the ranked value (ξ_j), and it is calculated as

$$a_j = \sum_{i=1}^{n} w_i 1(\xi_i < \xi_j), \quad (9)$$

where $1(\cdot)$ is the indicator function which is 1 when the condition is true and 0 otherwise, w_i is the ith weight, and ξ_i and ξ_j are the ith and jth values to be ranked, respectively. The second item b_j is designed to deal with elements with the same value. Considering the weight, for n tied elements with the same value, the vector of their tied ranks is $\mathbf{v} = [a_j + w_1, a_j + w_1 + w_2, \ldots, a_j + \sum_{k=1}^{n} w_k]^T$. Nevertheless, if we still take the mean of $\mathbf{v}$ to calculate the ranks of these tied elements (i.e., Eq. 7), the ordering of their weights will have a significant impact on the above results. To correct this deviation,

$$b_j = \frac{n+1}{2}\bar{w}_j \quad (10)$$

is adopted as the overall mean of all possible permutations of the weights of n tied elements, where $\bar{w}_j$ is the mean weight of these tied elements with the ranks

$$rank_j = a_j + \frac{n+1}{2}\bar{w}_j. \quad (11)$$

It is evident if w_j=1 for all j, Eq. 8 is equivalent to Eq. 11.

Through the above Eqs. 8, 9 and 10, one can calculate the weighted rank vectors of the ideal PSM and the tested PSM, and then obtain the corresponding WSpearman using Eq. 1. To our knowledge, how to calculate the WSpearman remains an open question in password research, and see why we choose the above method in Appendix C of the full paper.