2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks

fuzzyPSM: A New Password Strength Meter Using Fuzzy Probabilistic Context-Free Grammars

Ding Wang
Peking University
wangdingg@pku.edu.cn

Debiao He
Wuhan University
hedebiao@163.com

Haibo Cheng, Ping Wang
Peking University
{chenghaibo,pwang}@pku.edu.cn

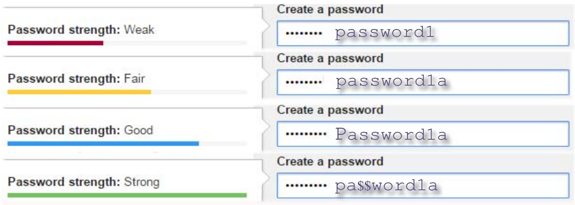
***Abstract*—To provide timely feedbacks to users, nearly every respectable Internet service now imposes a password strength meter (PSM) upon user registration or password change. It is a rare bit of good news in password research that well-designed PSMs do help improve the strength of user-chosen passwords. However, leading PSMs in the industrial world (e.g., Zxcvbn, KeePSM and NIST PSM) are mainly composed of simple heuristic rules and found to be highly inaccurate, while state-of-the-art PSMs from academia (e.g., probabilistic context-free grammar based ones and Markov-based ones) are still far from satisfactory, especially incompetent at gauging weak passwords. As preventing weak passwords is the primary goal of any PSM, this means that existing PSMs largely fail to serve their purpose.**

To fill this gap, in this paper we propose a novel PSM that is grounded on real user behavior. Our user survey reveals that when choosing passwords for a new web service, most users (77.38%) simply *retrieve* one of their existing passwords from memory and then *reuse* (or slightly modify) it. This is in vast contrast to the seemingly intuitive yet unrealistic assumption (often implicitly) made in most of the existing PSMs that, when user registers, a whole new password is constructed by mixing segments of letter, digit and/or symbol or by combining n-grams. To model users' realistic behaviors, we use passwords leaked from a less sensitive service as our *base dictionary* and another list of relatively strong passwords leaked from a sensitive service as our *training dictionary*, and determine *how mangling rules are employed* by users to construct passwords for new services. This process automatically creates a *fuzzy* probabilistic context-free grammar (PCFG) and gives rise to our fuzzy-PCFG-based meter, fuzzyPSM. It can react dynamically to changes in how users choose passwords and is evaluated by comparisons with five representative PSMs. Extensive experiments on 11 real-world password lists show that fuzzyPSM, in general, outperforms all its counterparts, especially accurate in telling apart weak passwords and suitable for services where online guessing attacks prevail.

I. Introduction

With so much of our lives digital and online, it is of great importance for web services to prevent our digital assets from unauthorized access. Due to their easiness to understand, convenience to use and low cost to deploy, textual passwords have been almost exclusively employed to protect access to our data since the dawn of the Internet, despite their many and known weaknesses [1]–[3]. Over the past few years, dozens of alternative authentication mechanisms (e.g., graphical passwords [4] and multi-factor authentication [5]) have been suggested, yet passwords stubbornly survive and reproduce with every new web services. This situation is largely due to the fact that each of these alternatives has its own prominent weakness(es) as compared to passwords [6]. What's more, due to the stickiness of user habits and the obscure transition costs [7], incremental improvement in security and usability often is insufficient to reach the activation energy to replace passwords. Consequently, passwords are likely to continue dominating the Internet in the foreseeable future.

It is well known that common users tend to choose weak passwords and render their accounts at high risks. To overcome this issue, nearly every respectable web service now enforces a password creation policy that requires user-chosen passwords to meet some composition rules, by mandating a list of rules and to reach certain strength threshold by imposing a password strength meter (see Gooogle's meter in Fig. 1).[1] A number of studies have investigated the impacts of password creation polices. In 1999, Adams [8] found that, when forced to comply with password creation policies that are incompatible with user work practices and organizational strategies, users generally attempt to circumvent such policies. In a lab study, Proctor et al. [9] measured the creation time, memorability, and crackability of passwords under various password creation policies, and found that stricter policies make passwords more resistant to cracking, but also harder to create and remember. Subsequent studies (e.g., [10], [11]) also confirm that if policies are not properly designed, users will adopt coping strategies that can impair both security and productivity.



Password strength: Weak
Create a password
•••••••• password1

Password strength: Fair
Create a password
•••••••• password1a

Password strength: Good
Create a password
•••••••• Password1a

Password strength: Strong
Create a password
•••••••• pa$$word1a

Fig. 1. Google's password strength meter and its real-time feedbacks.

In 2012, Ur et al. [12] revealed that only those password strength meters (PSMs) that provide accurate feedbacks about the strength of user-submitted passwords can lead to significant increases in password strength. In 2015, they further observed that a large fraction of users have misconceptions about the real security that their selected passwords can provide [13]. This is due in large part to the misleading feedback of the PSMs that are currently widely deployed in the wild (see [14], [15]). Besides Google's meter as shown in Fig. 1, for instance, one can confirm himself that the PSM of Yahoo measures `password` as "weak", `password1` as "medium" and `password123` as "strong"; the PSM of the World Wide Web Consortium (W3C), which advertises itself as "an international community that develops open standards to ensure the long-term growth of the Web", measures `password` as "very weak (8%)", `password1` as "weak (26%)", `Password1` as "sufficient (46%)" and `Password123` as "strong (67%)". Such inaccurate feedbacks would lead common users to hold a

[1] https://accounts.google.com/SignUp

978-1-4673-8891-7/16 $31.00 © 2016 IEEE
DOI 10.1109/DSN.2016.60

belief that adding a digit or capitalizing the first letter of their weak passwords will yield secure ones.

Accurate measurement of password strength necessitates a proper metric. Perhaps the most influential metric is the entropy-based approach proposed in NIST Electronic Authentication Guideline SP-800-63 [16]. The NIST PSM, as admitted in [16], is essentially an ad hoc estimation of password strength based on heuristic rules (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"). Most of the high-profile PSMs over the Internet (e.g., Yahoo, Gmail, Microsoft and Paypal) "perfectly" [15] capture the spirit of the NIST PSM. However, Weir et al. [17] and Kelley et al. [18] illustrated that this entropy-based approach only provides a very rough approximation of password strength as measured in terms of real-world guessing resistance.

Recent research (e.g., [12], [18]) shows that a more effective metric for password strength is "guessability", which relates to real-world security and characterizes the time complexity required for a password-cracking algorithm to recover an account. This is generally achieved by measuring *the guess number* (i.e., search space size) required to break that password [19]. This measure eliminates the dependence on specific natures of the underlying authentication system and the attacker, and it is only related to the attack technique and to the complexity of user password. It is particularly desirable for fair comparisons of different cracking algorithms against a given dataset and of different passwords against a given cracking algorithm. Consequently, guessability-based metric has been widely preferred in latest works (e.g., [14], [20], [21]).

Evaluating the "guessability" of passwords dovetails with recent advances in password guessing techniques. According to whether involving the authentication server, guessing attacks can be classified into: online guessing and offline guessing (see [22]); According to whether involving user-specific information, guessing attacks can be classified into: trawling guessing and targeted guessing (see [14]). This work mainly focuses on trawling guessing, while other attacking vectors (e.g., targeted guessing [14], keylogging [23] and shoulder surfing [24]) and their countermeasures are out of our scope. Hereafter whenever the term "guessing" is mentioned, it means *trawling* guessing.

Trawling guessing continues to pose a serious threat to user password-protected accounts. More specifically, though the problem of offline guessing can be partially relieved by using salt and sophisticated hash (e.g., bcrypt and PBKDF2), an attacker's guessing efficiency would be orders of magnitude higher than expected by using dedicated hardware and software (see [25]). In addition, it is likely that the litany of recent leakages of tens of millions of plain-text passwords (e.g., 32 million RockYou [26] and 30 Millon Tianya [27]) would further improve an attacker's guessing efficiency, because it enables the attacker to gain deeper insights about human habits in selecting passwords. For instance, such datasets have recently been exploited by probabilistic context-free-grammar (PCFG) based [28] and Markov-Chain-based [29] cracking algorithms as training sets to reveal user password behavior. Similarly, though the threat of online guessing can be partially addressed by using modern machine-learning-based detection, rate-limiting and lockout strategies (e.g., [30], [31]), the attacker's capabilities would be enhanced by exploiting password leakages, in contrast to previous brute-force or dictionary-based approaches. When further considering the prevalence of botnets [32] that can offer enormous computational and bandwidth resources to attackers, both offline and online guessing will be more damaging.

To more accurately capture attackers' advanced guessing strategies and provide more accurate password strength estimation, Castelluccia et al. [33] for the first time developed an adaptive PSM that is based on the Markov-Chain-based probabilistic guessing model. Their Markov-based PSM can react dynamically to changes of user password behavior and is shown to be much more accurate than the PSMs of NIST, Google and Microsoft. Almost at the same time, Houshmand and Aggarwal [34] proposed an adaptive PCFG-based PSM, which can suggest better password candidates if the strength of a user's original password is below the allowed threshold. However, this PCFG-based PSM has neither been evaluated on real-life password datasets to show its accuracy nor compared with other PSMs to show its pros and cons.

In 2015, Carnavalet and Mannan [15] studied twenty-two PSMs that are currently deployed at popular web services and password managers. They found that most meters are based on ad hoc design, highly inconsistent outcomes are given for the same password by different meters and many weak passwords are labeled as "strong". Comparatively, the PSMs of Dropbox (called Zxcvbn [35]) and Keepass (called KeePSM [36]) perform the best, and these two are also the only ones that provide some explanations for their design choices.

A. Motivations

Surprisingly, as far as we know, no state-of-the-art PSMs from academia (i.e., Markov-based PSM [33] and PCFG-based PSM [34]) has ever been adopted into leading web services or password managers. This conclusion is based on our investigation into the top 120 sites in the Alexa Global Top 500 sites list based on their traffic ranking (http://www.alexa.com/topsites) as well as on the results about 50 meters studied in [14] and about 22 meters studied in [15]. This can be largely attributed to the fact that *there is little evaluation or comparison* of state-of-the-art PSMs from academia (i.e., Markov-based PSM [33] and PCFG-based PSM [34]) with the best PSMs from industry (i.e., Zxcvbn [35] and KeePSM [36]). As such, the advantages and disadvantages of these PSMs remain subtle and unknown to the security community, thereby PSM adoption and migration are unlikely.

Besides, to the best of our knowledge, *Markov-based PSM [33] and PCFG-based PSM [34] have never been publicly compared with each other*, and it remains an open question as to which performs better. Castelluccia et al. [33] conjectured that the Markov models could be used to create better PSMs because that "previous work (e.g., [19]) has already shown that Markov-model based password crackers outperform existing cracking techniques (such as PCFG-based ones)". However, as we will show, this is not the case and just on the contrary, PCFG-based PSM generally outperforms Markov-based PSM.

Another unsatisfactory aspect of these PSMs from academia is that, they all are *implicitly* based on the underlying assumption that *users will build new passwords from scratch for a new service* by combining segments of words, digits and/or symbols (i.e., the PCFG-based approach [34], [37]) or by combining Markov n-grams (i.e., the Markov-based approach

[33]). Actually, this is not true in practice. As revealed in our online survey with 442 participants, when registering for a new service, *most users (77.38%) simply retrieve one of their existing passwords from memory and then reuse (or slightly modify) it.* Only 14.48% of users would construct a completely new password from scratch. These two figures are well consistent with those of the survey with 224 participants conducted by Das et al. [38]. Thus, it is crucial to see whether PSMs can be built on this new observation.

Last but not the least, existing PSMs only show their accuracy on few English password datasets. More specifically, the Markov-based PSM [33] experiments on two datasets, one from the social forum Rockyou [26] and another from the programmer forum Phpbb [39], while the PCFG-based PSM [34] merely shows its feasibility by using the Rockyou dataset (and has not shown its accuracy). *On the one hand, it is more desirable that passwords from other more sensitive types of services (e.g., e-commerce and dating) can be tested. On the other hand, as the existing PSMs are primarily designed for measuring English passwords, whether they can be used for measuring non-English passwords?* At the time of this writing, there are 668 million Chinese netizens over the Internet [40], which account for over a quarter of the world's total Internet population, and thereby it is interesting to explore the applicability of existing PSMs to Chinese web passwords.

B. Our contributions

In this work, we make the following key contributions:

- **A user survey**. To reveal user behavior in selecting passwords for new services, we conduct a user survey with 442 effective participants, the largest one that's ever conducted for this purpose and the first one that's conducted on Chinese netizens. Our results accord with previous survey results [38] on English users. This survey sheds light on how common users employ mangling rules to modify their existing passwords for new online accounts and on how we should measure password strength.
- **A new meter**. To model users' realistic behaviors, we use one dictionary of relatively weak passwords leaked from a less sensitive service as our *base dictionary* and another dictionary of passwords leaked from a sensitive service as our *training dictionary*, and determine how mangling rules are employed by users to construct passwords for new services. This process automatically creates a fuzzy probabilistic context-free grammar (PCFG) and gives rise to our fuzzy-PCFG-based meter, fuzzyPSM. It can react dynamically to changes in how users choose passwords, and extensive empirical comparisons with five representative PSMs show that fuzzyPSM generally performs the best: It takes the first place in gauging weak passwords, while being second in gauging strong passwords.
- **A systematic evaluation.** We perform a series of experiments and use two different rank correlation metrics to investigate the effectiveness of five state-of-the-art PSMs: two from academia (i.e., Markov-based [33] and PCFG-based [34]), two from industry (i.e., Zxcvbn [35] and KeePSM [36]) and one from the standards bodies (i.e., NIST PSM [16]). Our evaluation builds on 11 large-scale real-world password lists, which consist of a total of 97.43 million passwords, cover various popular Internet services and are the largest corpus ever collected for PSM evaluation.
- **Some insights.** We derive a number of insights, some expected and some surprising, from our extensive experiments. In most cases, PCFG-based PSMs outperform Markov-based ones, which is opposed to common belief (as hold in [33]). As expected, in all cases academic PSMs outperform PSMs from the industrial world. The results show that PSMs originally designed for English users can be used for non-English users if training sets are properly chosen.

Roadmap. Section II provides some preliminaries. Section III details our user survey. Section IV elaborates on our meter FuzzyPSM. Section V focuses on the experimental methodologies and the evaluation results. Finally, Section VI concludes the paper.

II. Preliminaries

We now explicate the security model that underlies our and existing meters, provide a formal definition for the practically ideal meter and introduce the metrics to evaluate meters.

A. Security model

As said earlier, the objective of PSMs is to quantify *how much efforts that an attacker has to invest* before guessing the correct password. To this end, one must first make clear which types of attacks the attacker can launch, because different kinds of attacks often involve quite different resources and guessing strategies. As with most related works [19]–[21], [29], [33], [34], [37], in this work we mainly focus on trawling guessing attacks,[2] while targeted guessing attacks [14] are not considered, because this latter kind of attack involves user-specific data and it still remains an open question as to how the attacker would exploit such personal data. In addition, other attack vectors such as keylogging [23] and shoulder surfing [24], where the attacker obtains the password by non-cryptographic ways, are unrelated to password strength and therefore outside the scope of this work.

In trawling guessing attacks, the attacker does *not* care who is the victim and her only aim is to find out the password that matches *any one* of the accounts (see [41]). Consequently, her best strategy is to repeatedly make guesses of *the most likely password first* (i.e., to try the guesses in decreasing order of probability). Trawling guessing attacks can be further classified into online trawling guessing and offline trawling guessing. In an on-line trawling guessing attack, the attacker tests the correctness of her guesses by attempting to interact with the authentication server. In an off-line trawling guessing attack, the attacker has gained direct access to the hashed password and to crack it, she does *not* need to interact with the server and can *offline* hash her guessing password using the same hash algorithm (e.g., scrypt or PBKDF2)[3] and offline compare it with the target. If these two hashes match, the attacker has found the original password.

[2] Though the security model in most PSMs (e.g., [33]–[37]) and related literature ([19]–[21], [29]) assumes a trawling guessing attacker, yet in contrast to this work, such an assumption is often only *implicitly* made. This is highly likely to lead misconceptions for security administrators.

[3] Though it has long been recommended that sites shall store passwords in salted-hash by using dedicated hash functions (e.g., scrypt and PBKDF2), the reality is that 64% of the leaked datasets are in clear-text or in MD5 without a salt [42], and even 11 of the top-500 sites [43] store passwords in clear-text.

TABLE I. Comparison of different guessing attacks

Guessing attack types		Use personal data	Interact with server	Major constraints for attacker	# of guesses allowed	Considered in this work
Trawling	Online	No	Yes	Detection, lockout	$< 10^4$	Yes
	Offline	No	No	Attacker power	$> 10^9$	Yes
Targeted	Online	Yes	Yes	Detection, lockout	$< 10^4$	No
	Offline	Yes	No	Attacker power	$> 10^9$	No

Table I shows that the most critical difference between an online and an offline attack is the number of guesses that the attacker can make. In an online attack, the system that the attacker attempts to attack is operational, and security mechanisms such as suspicious login detection [30] and lockout policies [31], are still active. Thus, generally an online attacker can only make a few guesses before getting locked out (e.g., 100 failed login attempts in 30 days as suggested by NIST [16]), and thus the best strategy for her is to try these few most popular passwords [41], [44]. On the other hand, an offline attacker is not limited by the defender's security mechanisms and is allowed as many guesses as she can made, while the only constraints are her own time and computational resources. This means that an offline guessing session may consist of trillions of guesses [21], [45], and thus an offline attacker can try not only popular passwords but also very rare ones as her guesses.

All in all, a trawling guessing attacker's best guessing strategy is always that *more popular passwords are tried earlier*, and *the number of guesses that needs to be tried before guessing the correct password* well characterizes the efforts that an attacker has to invest. This number can be defined as the target password's strength.

B. Formal definition for the practically ideal meter

To measure a given password pw's strength, PSMs generally do not directly output pw's guess number. Instead, they output each guess's entropy (e.g., NIST [16]) or probability that is associated with the derivation of this guess (e.g., probabilistic-model-based ones [33], [34]). Based on the values of entropy or probability, all the guesses can be sorted and pw's guess number can thus be definitely determined.

Password. The formal definition of (textual) password was first given by Ma et al. in IEEE S&P 2014 [29]. A password pw is a string of characters confined to the alphabet Σ, with its length between L_{min} and L_{max} [29]. Therefore, the set of passwords that are allowed in an authentication system is:

$$\Gamma = \bigcup_{l=L_{min}}^{L_{max}} \Sigma^l \tag{1}$$

Generally, the alphabet Σ can be set to any subset of the 95 printable ASCII characters. However, since these 95 characters are typically categorized into the four groups: digits, lower-case letters, upper-case letters and special symbols, Σ is often set to include any combination of these four groups, such as the 10 digits and the 36 lower-case letters with digits. We investigate top 50 sites and find that, 11 of them require $L_{min} = 6$ and $L_{max} = 20$, and 7 of them require $L_{min} = 6$ and $L_{max} = 16$. Note that, in the cracking experiments of this work, we set Σ to include the full 95 characters.

Password strength meter. A password strength meter (meter for short) is a function $M(\cdot)$ that takes as input a password pw over an alphabet Σ and outputs a probability (a real number) in [0, 1]:

$$M : pw \rightarrow [0, 1], \text{ where } \quad pw \in \Gamma \tag{2}$$

The stronger the password pw is, the lower the probability $M(pw)$ will be. Note that, only when $M(\cdot)$ is a probabilistic-model-based PSM, $M(\cdot)$ satisfies that: $\sum_{pw \in \Gamma} M(pw)$ =1. For instance, the PSMs in [33], [34] are two probabilistic-model-based ones, while Zxcvbn [35], KeePSM and NIST PSM [16] are not. The strength $M(pw)$ for each password pw constitutes an estimation of the amount of work that an adversary is required to invest in order to succeed. In practice, the values of a meter are often grouped into a few buckets, such as "[`weak`, `medium`, `strong`]" (see Apple's meter) and "[`weak`, `fair`, `good`, `strong`]" (see Google's meter). Note that a meter can be either mandatory or just suggestive. The former kind of meters require that only a password pw with its probability $M(pw)$ below the pre-defined threshold can be accepted, while the later kind of meters merely provide users with information about the strength of their passwords.

Ideal meter. Essentially, the ultimate goal of a probabilistic PSM is to *reproduce* the probability distribution $\mathcal{X}$ of the passwords in a target authentication system. Theoretically, what an ideal meter $M(\cdot)$ can achieve is that,

$$M(pw) = p_{pw}, \quad \forall\ pw \in \Gamma \tag{3}$$

where p_{pw} is the *true probability* of the password pw drawn from $\mathcal{X}$. In other words, an ideal meter can reproduce *exactly* the distribution $\mathcal{X}$. In practice, it is virtually impossible to determine the value of p_{pw}. Fortunately, for popular passwords (e.g., with $f_{pw} \geq 4$), one can use the *empirical probability* $f_{pw}/|\mathcal{DS}|$ to approximate p_{pw} with a relative standard error about $1/\sqrt{f_{pw}}$ [41], where f_{pw} is the frequency of pw in the password dataset $\mathcal{DS}$ (which is sampled from $\mathcal{X}$). This means that one can draw a *large* sample $\mathcal{DS}$ from $\mathcal{X}$ and sort $\mathcal{DS}$ in decreasing order of empirical probabilities, then the *popular part* of the list can be used as an ideal meter with assured accuracy: the order in the list is just the guess number (i.e., strength) of the password to be measured.

Accordingly, any real-world PSM (e.g., [33], [34]) can be seen as an approximation of this ideal meter. We then can evaluate the goodness of a real-world PSM by measuring the differences in their outcomes: 1) the differences in probability for every password; and 2) the differences in guess number for every password. We note that the first kind of difference not only is cumbersome to measure, but also can be well characterized by the latter kind of difference if the focus of a PSM is its accuracy for guess number estimation (see Section II-A). As an example, suppose two meters $M_1(\cdot)$ and $M_2(\cdot)$, let $M_1(\cdot)$ be an ideal meter and $M_2(\cdot)$ outputs as follows:

$$M_2(pw_i) = \begin{cases} M_1(pw_1) + (M_1(pw_2) - M_1(pw_3))/2, \\ M_1(pw_2) - (M_1(pw_2) - M_1(pw_3))/2, \\ M_1(pw_i), \quad \text{for all } \ i \geq 3 \text{ and } pw_i \in \Gamma \end{cases} \tag{4}$$

One can confirm that $\sum_{pw_i \in \Gamma} M_2(pw_i)$ =1 and that, if $M_1(pw_1) \geq M_1(pw_2) \geq M_1(pw_3) \geq \cdots$, then $M_2(pw_1) \geq M_2(pw_2) \geq M_2(pw_3) \geq \cdots$. This means that $M_2(\cdot)$ is a probabilistic-model-based meter and that $M_2(\cdot)$ can produce *the same guess number* for a given password as the ideal meter $M_1(\cdot)$ does. Thus, their behaviors are indistinguishable under our security model. This gives rise to the following relaxed (yet more practical) ideal meter.

A practically ideal meter. We say a meter $M(\cdot)$ is a practically ideal meter if it can output the same guess number for a given password as the ideal meter does. Formally, $M(\cdot)$ satisfies that, whenever $p_{pw_i} \geq p_{pw_j}$,

$$M(pw_i) \geq M(pw_j), \text{ for all } pw_i, pw_j \in \Gamma \qquad (5)$$

where p_{pw} is the true probability of the password pw drawn from the underlying distribution $\mathcal{X}$. In terms of guess number, $M(\cdot)$ is distinguishable from an ideal meter. The prominent advantage of this definition is that, it enables us to focus on the guess number and to use well studied metrics (see Section II-C) to accurately measure the defects of any realistic meter.

We note that Castelluccia et al. [33] also defined an "ideal meter", yet one can confirm that, actually, their ideal meter is not an *ideal* one because by their definition, it can not reproduce the probability distribution of the passwords in a target authentication system. Their ideal meter is essentially some ad hoc examples of our above practically ideal meter.

C. Two rank correlation metrics

According to the above definitions, to measure the goodness of a PSM $M(\cdot)$, one can measure its distance from a practically ideal meter. This can be achieved by measuring the correlation between the ranked guess numbers (or equally, probabilities) that are produced by each meter. Since the probabilities produced by PSMs are generally highly skewed (see Fig. 2 of [29]), non-parametric rank correlation metrics shall be used. Spearman coefficient ρ [46] and Kendall τ [47] are two such metrics that have been most widely used. Both ρ and τ are real numbers in $[-1, 1]$, with 1 indicating the agreement between the two rankings is perfect, -1 the worst and 0 independent. More details are referred to the full version (see http://t.cn/RG8Ewf3) of this work .

III. User survey

To obtain more accurate assessment of the strength of user-chosen passwords, we need to first better understand user behaviors, i.e., how users create passwords when registering a new account. To this end, we conduct an anonymized user survey by using Sojump (sojump.com), the most popular Chinese online survey platform. A copy of our questions is available at http://www.sojump.com/jq/6443561.aspx, and the third part relates to this work. For better comparison of Chinese user behaviors with their English counterparts (see the US-located survey in [38]), some of our questions are specifically designed to be as close to [38] as possible.

We optimize the survey questions before release to minimize user refusal and dropout (see [11]) by iteratively getting feedbacks from students in other teams of our lab. Finally, a few potentially offensive questions (e.g., "How many PIN codes do you maintain?" and "If you reuse an existing password for this new email account, which category is it likely to come from?") are abandoned and our survey takes about 22 to 35 minutes. Our team members are each requested to invite 10 to 20 reliable friends, relatives or colleagues to complete the survey. We further ask some of the participants who respond very positively to our invitation (e.g., they think our survey are of important value to the society) to distribute the survey, and they are also asked to notify us the number of invitations they send. Altogether, a total of 983 invitations are sent and we get 442 effective responses. We first elaborate on the survey results and then report the demographics of respondents.

We started our survey by specifying a scenario where the participants are asked to create an email account that will be frequently used, and issued a series of questions regarding their password choices. Our first question aims to figure out how similar their password of this new account is to that of their existing accounts. Fig. 2 shows that 77.38% of users would reuse or simply modify an existing password. This value is inconceivably consistent with that of English users (77% [38]). While there are 6.2% less Chinese users than English users in reusing an existing password, there are 14.86% more English users than Chinese users in creating an entirely new password. In 1999, Adams and Sasse [8] found that 50% of their 139 questionnaire respondents employ their own methods for creating memorable multiple passwords through transformation of existing passwords. This potentially implies that, as time goes on (and the number of web accounts one has to maintain constantly increases), the degree of reusing passwords are becoming more serious.

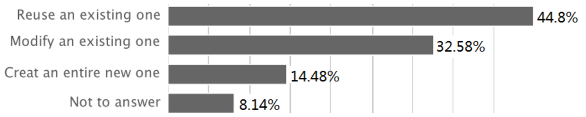


Fig. 2. What will you do when requested to create passwords for a new email account that will be frequently used?

If a user modifies an existing password, it remains a question as to how similar the modified one is to the original one. Thus, we ask a related question as shown in Fig. 3. We can see that the answers "very similar" and "the same" at least account for 61.77%, and "similar" account for another 20%. This indicates that, over 80% of users will submit a password for the new email account that is similar to their existing passwords. This observation is in vast contrast to the seemingly intuitive yet unrealistic assumption (often implicitly) made in most of the existing PSMs that, when user registers, a whole new password is constructed by mixing segments of letter, digit and/or symbol (see PCFG-based PSM [34]) or by combining n-grams (see Markov-based PSM [33]). This has great implications for measuring password strength: as the majority of users' passwords for the new account would be similar to their existing passwords, and there is high potential that attacker has already learnt the latter (e.g., through unending password leakages [26], [27], [48]), and thus *the strength of password for the new account shall mainly relate to how much uncertainties are introduced during the modification process*. This inspires our fuzzyPSM in Sec. IV.

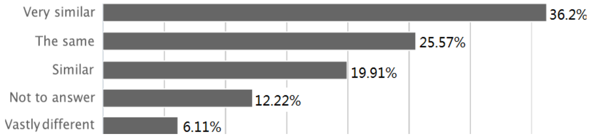


Fig. 3. How you describe the similarity of your password for the new account as compared to that of your existing accounts?

Fig. 4 helps us to understand why users modify (but not reuse) an existing password. As expected, 51% of users aim to increase security, which accords with the observation that users often believe that simple modifications would greatly increase strength of their passwords — "I added '!' at the end to make it secure" [13]. In contrast, this figure for English users is only 24% [38]. A plausible reason may be that, as there are much more high-profile Chinese web services than English ones have leaked user accounts in the past few years [27], [29], Chinese users are more concerned with the current security of password-based authentication. The figures for fulfilling password policies and improving memorability are 42.76% and 32.58%, respectively, which are comparable to that of English users as reported in [38].

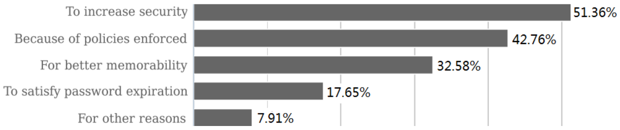


Fig. 4. If you modify an existing password for the new account, why (multiple-choice)? It seems the majority of users aim to increase security.

The response to which transformation rules are employed by users to modify their existing passwords is summarized in Fig. 5. Evidently, concatenation (e.g., "adding a digit/symbol at the beginning/end") takes the lead, followed by "capitalization" and "leet" (e.g., `a`↔`@` and `4`↔`for`). In addition, "substring movement", "reverse" and "add site-specific info" also have their places. Though these figures cannot be directly compared with [38] because our question is not single-choice, they largely accord with findings about English users.

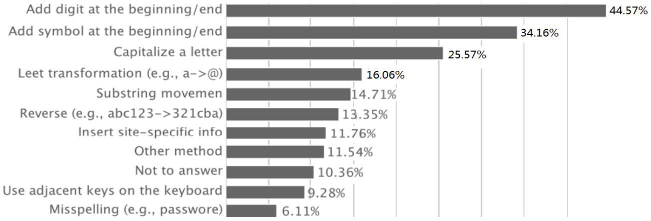


Fig. 5. What transformation rules do you use when modifying your existing passwords (multiple-choice)? The majority of users prefer concatenation.

In the following few questions we investigate where these above transformation rules happen in the password. Figs. 6 and 7 show that users love to place digit/symbol at the end, middle and beginning in decreasing order of likelihood. This roughly accords with English users [38]. As for capitalization, Fig. 8 shows that, 47.96% of Chinese users tend to perform capitalization at the beginning of password, while this figure for English users is quite similar (44%). However, 22.62% of Chinese users choose not to use the transformation rule of capitalization, while this figure for English users is only 6%. These results match our observations in leaked real-life password datasets (see Table VIII).

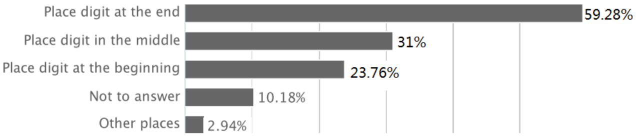


Fig. 6. If the site requests that the password shall include a digit, where will you usually place the digit? (multiple-choice)

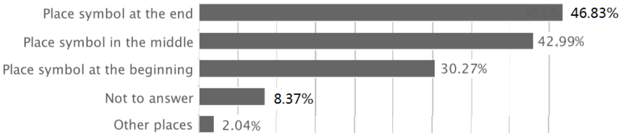


Fig. 7. If the site requests that the password shall include a symbol, where will you usually place the symbol? (multiple-choice)

We also obtained some basic demographic information about our participants: two-thirds are male; 80.55% are between 18∼34 years old, and 15.67% are older than 35; 80.55% are pursuing or with at least a bachelor's degree, and 43.44% are pursuing or with at least a master's degree. This is as expected, for the majority of our team members are young PhD candidates and thus their personal relationships are mainly young and educated people. More detailed information are referred to [45]. Comparatively, our participants are more diversified than that of [38], in which 93% of its participants are 18∼34 years old and 92% have at least a bachelor's degree. Nevertheless, our participants are younger and more educated than the common population, and they may be more security-conscious and technically savvy. Hence, it is likely that we are underestimating the problem.

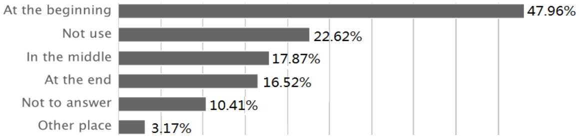


Fig. 8. If your new password involves capitalization, where will it happen (multiple-choice)? About half of the users capitalize the first letter.

Ethics and limitations. To the best of knowledge, this survey is the first one that targets Chinese users, the world's largest Internet population. We got approval from our center's IRB for this survey. The participants are surveyed anonymously and the results are all aggregated such that no user identifiable information can be inferred. Though our participants are superior to earlier related studies (on English users [38], [49], [50]) in terms of number and diversity, yet our survey is subject to the inherent limitation of any password-related survey: users may provide false data and the ecological validity is hard to be assured. Still, in many aspects our results comply with real-life datasets (see Sec. V-B) and with English users (e.g., the rate of password reuse and the location of adding digit/symbol).

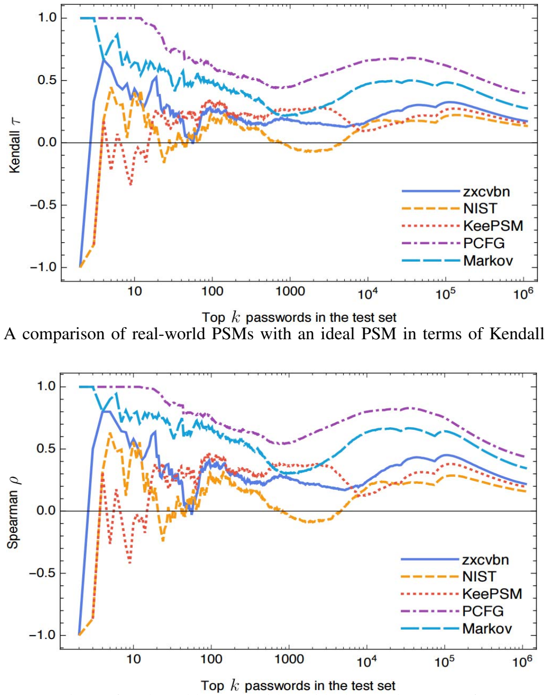
IV. Our proposed meter fuzzyPSM

We now describe fuzzyPSM, a new PSM that is based on fuzzy PCFG. To explain why our meter prefers PCFG-like model but not Markov-like model, we first provide a brief comparison of existing PSMs and show their main limitations.

A. A brief comparison of existing PSMs

To the best of our knowledge, the two state-of-the-art PSMs (i.e., Markov-based PSM [33] and PCFG-based PSM [34]) have never been publicly compared with each other, and it remains an open question as to which performs better. Castelluccia et al. [33] conjectured that the Markov models could be used to create better PSMs because that "previous work (e.g., [19]) has already shown that Markov-model based password crackers outperform existing cracking techniques (such as PCFG-based ones)". Indeed, a number of very recent studies (e.g., [21], [29], [45]) also confirm the superiority of Markov-based model in password cracking. However, as we will show here (and in Sec. V), this trend does not hold in the case of gauging password strength, and just on the contrary, PCFG-based PSM generally outperforms Markov-based PSM.

As a concrete example, we use these two machine-learning-based PSMs to measure 6.43 million CSDN passwords which were disclosed from csdn.net, a popular community for software developers in China. Generally, the effectiveness of a machine-learning-based meter depends on two factors: the algorithm itself and the training sets used. To preclude the impacts of training sets, as with [17], [33], we randomly split the CSDN dataset into equally four parts, and use part-1 for training and part-4 for testing. As both PSMs have no open-sourced codes, we implement the PCFG-based PSM [34] and Markov-based PSM [33] according to the most recent improvements in [29]. More specifically, for PCFG-based PSM the probabilities associated with letter segments are learned directly from the training process, and for Markov-based PSM we use the backoff approach. To ensure the correctness of our implementations, we used the guesses[4] output by these two PSMs to repeat the cracking experiments in [29], [45] and the cracking results are in full accord. We plan to open-source all the codes in this work to facilitate the research community.



(a) A comparison of real-world PSMs with an ideal PSM in terms of Kendall τ

(b) A comparison of real-world PSMs with an ideal PSM in terms of Spearman ρ

Fig. 9. A comparison of five leading PSMs in terms of Kendall and Spearman coefficients (1/4 CSDN passwords for training and another 1/4 for testing).

Then we adopt the two rank correlation metrics (i.e., Kendall τ and Spearman ρ) to measure the distance of guess numbers output by a real-world PSM to guess numbers output by the ideal meter (see Sec. II). This gives rise to Figs. 9(a) and 9(b), respectively. Note that, a value k on the x axis indicates that the data point in the plot is calculated for the *set* of the top $1, 2, \cdots, k$-ranked passwords. Both metrics provide nearly the same results, and their similarity holds in all our later experiments. Comparatively, the Kendall τ metric is more sensitive to minor differences, and thus hereafter we only provide Kendall τ based results to save space. In addition, here we also provide comparisons of leading PSMs from industry (i.e., Zxcvbn [35], Keepass PSM [36] and NIST PSM [16]), which are all rule-based but not machine-learning-based PSMs.

Both metrics show that PCFG-based meter performs best among existing PSMs, and the three rule-based PSMs from industry are inferior to the two machine-learning-based PSMs from academia. While the later finding is expected, the former is somewhat surprising. *This invalidates the widely believed conjecture* that Markov-based model "could be used to create better proactive password strength meters (than PCFG)" [33]. *This is quite un-expected and counter-intuitive*, because Markov-based model has been widely shown better than PCFG-based model in password cracking (see [20], [29], [45]).

B. Explication of the unexpected superiority of PCFG PSM

To explicate this seemingly paradoxical observation, we provide a microscopic grasp of the differences between PCFG-based PSM [34] and Markov-based PSM [33]. More specifically, we examine the differences in the guess numbers output by each PSM for the testing passwords. In Fig. 10, each *red* data point stands for a password from the training set. Its co-ordinate (x_i, y_i) means that this password's strength can withstand x_i guessing attempts measured by the ideal meter and withstand y_i guessing attempts measured by the *PCFG-based* meter. Similarly, each *blue* data point stands for a password from the same training set and its co-ordinate (x_i, y_i) represents the guess numbers given by the ideal meter and *Markov-based* meter. Note that, a value k on the x axis indicates that the data point in the plot is calculated for the *set* of the top $1, 2, \cdots, k$-ranked passwords. It is not difficult to see that, the better a meter is, the more closeness the data points will assemble the green line. This indicates that PCFG-based meter outperforms Markov-based meter, which is especially evident when measuring weak (top) passwords.

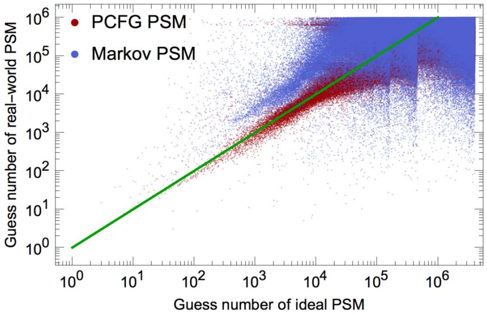


Fig. 10. A comparison of PCFG-based PSM with Markov-based PSM in terms of guess number (1/4 CSDN for training and another 1/4 for testing).

To gain a more concrete grasp of Fig. 10, we summarize in Table II the guess numbers given by each PSM for some typically weak passwords. The weaknesses of these six passwords are demonstrated by the small ranks in the training set (see column 2) and by small guess numbers given by the ideal meter (see column 3). In four of six cases, the PCFG-based meter outputs better guess number estimation than the Markov-based meter. In all, our fuzzyPSM, as further illustrated later, outputs the most accurate estimation of password strength.

Table III summarizes the number of un-usable guesses that are produced by PCFG- and Markov-based cracking models. A guess is called *un-usable* if it is produced by the guess model, yet it does not appear in the test set. Such un-usable guesses partially indicate the goodness of a cracking model: less un-usable guesses would mean a better model. One can see that, when the guess number is small, PCFG-based model produces less un-usable guesses, yet this situation reverses when the guess number is larger than 10^6. This explains why (at large guesses) Markov-based model generally cracks more passwords than PCFG-based model (see [20], [29], [45]). At this point, by using 6.43 million CSDN passwords

[4] Probabilistic-model-based PSMs are essentially password cracking tools [28], [33] and they can output guesses in decreasing order of probability.

TABLE II. THE GUESS NUMBER GIVEN BY EACH PSM FOR WEAK PASSWORDS (USING 1/4 CSDN PASSWORDS AS THE TRAINING SET).

Typical password	Training set	Ideal PSM	PCFG PSM	Markov PSM	fuzzyPSM
123qwe	10182	8527	771719	324	4764
123qwe123qwe	2778	3105	194671	236965	4505
password123	2315	2522	4219	6124	2397
Password123	27097	10170	29341	31009	23438
password	18	21	20	35	46
p@ssw0rd	76	259	337982	290	274

* A guess number with dark gray means it is the closest to the guess number given by the ideal PSM, and a guess number with light gray represents the second best one.

TABLE III. NUMBER OF UN-USABLE GUESSES THAT ARE PRODUCED BY PCFG- AND MARKOV-BASED CRACKING MODELS

	top 10^2		top 10^4		top 10^6		top 10^7	
	PCFG	Markov	PCFG	Markov	PCFG	Markov	PCFG	Markov
un-usable guesses	1	12	160	4509	372883	359401	9153626	8587614

as an example, we manage to reconcile the two seemingly paradoxical observations: the PCFG-based model is better at measuring passwords, yet the Markov-based model is better at cracking passwords. A plausible underlying rationale is that, the smoothing techniques (e.g., backoff, Laplace and Good-Turing [29]) used in Markov-based models make them better at cracking passwords (i.e., predicting more unseen passwords), yet this in turn makes Markov-based PSMs subject to the sparsity problem and worse at measuring weak passwords.

C. A meter based on fuzzy PCFG

Though the PCFG-based PSM performs best among the existing PSMs, yet it is still subject to inherent limitations. In the PCFG-based model, passwords are considered to be formed by three kinds of segments, i.e. L, D and S, with each standing for the string sequences of letters, digits and symbols, respectively. For instance, p@ssw0rd is composed of segments p, @, ssw, 0 and rd, with the base structure $\mathsf{L}_1\mathsf{S}_1\mathsf{L}_3\mathsf{D}_1\mathsf{L}_2$; Password123 is composed of segments Password and 123, with the base structure $\mathsf{L}_8\mathsf{D}_3$; 123qwe123qwe is composed of segments 123, qwe, 123, and qwe, with the base structure $\mathsf{D}_3\mathsf{L}_3\mathsf{D}_3\mathsf{L}_3$. The probability (strength) of "p@ssw0rd" is computed as P("p@ssw0rd")= $P(\mathsf{L}_1\mathsf{S}_1\mathsf{L}_3\mathsf{D}_1\mathsf{L}_2)\cdot$ $P(\mathsf{L}_1 \rightarrow \texttt{p})\cdot$ $P(\mathsf{S}_1 \rightarrow \texttt{@})\cdot$ $P(\mathsf{L}_3 \rightarrow \texttt{ssw})\cdot$ $P(\mathsf{S}_1 \rightarrow \texttt{0})\cdot$ $P(\mathsf{L}_2 \rightarrow \texttt{rd})$. In the 2009 original proposal [28], the probabilities for base structures, digit segments and symbol segments are learned from the training set by counting, while the probabilities for letter segments are computed by using an external input dictionary. Ma et al. [29] suggested that it would be better when the probabilities for letter segments are computed by also learning from the training set. This advice has been widely accepted [20], [29] and we also adopt it.

From the above examples of how the PCFG-based PSM measures passwords, it is not difficult to find that this kind of PSM essentially assumes that passwords created for a new service are built of L, D and S segments from *scratch*, and it primarily takes into account the concatenation rule. However, it disregards two basic facts: (1) an overwhelming percentage of users build passwords from *existing passwords* (see Fig. 2); and (2) other transformation rules like capitalization and leet are also popular (see Fig. 5). Though several other PSMs (e.g., [16], [33], [35]–[37]) have considered the second reality, yet so far most PSMs fail to consider the first one.

To model users' password reuse behaviors, we employ: (1) passwords leaked from a less sensitive service as our *base dictionary* $\mathcal{B}$ to construct a basic password parsing trie-tree; and (2) another list of relatively strong passwords leaked from a sensitive service as our *training dictionary* $\mathcal{T}$. Note that now our trie-tree only includes basic passwords that are lower-cased and with length no shorter than 3. Then, we parse each password in the training dictionary and learn *which and how mangling rules are employed* by users to construct passwords for new services. This process automatically creates a *fuzzy* probabilistic context-free grammar (PCFG) and gives rise to our fuzzy-PCFG-based meter, fuzzyPSM. Our meter includes three phases: training, measuring and update.

In the training phase, we measure the frequencies of certain patterns associated with training passwords. We assume all passwords are built using segments from the basic password set $\mathcal{BS}$= $\{b_1, b_2, \cdots, b_N\}$, and $\mathcal{B} \subseteq \mathcal{BS}$. We denote a*basic password* of length n to be with base structure B_{n}, and consider the three most popular transformation rules (i.e., concatenation, capitalization and leet). Each password from the training set $\mathcal{T}$ is parsed by the trie-tree using longest prefix match. Other more sophisticated parsing methods (e.g, Bayesian parsing [51]) may be used to improve accuracy, but we in this work keep our meter as simple as possible and show that, even this simple meter is able to provide accurate estimations.

TABLE IV. PROBABILITY OF BASE STRUCTURES AND SEGMENTS

LHS	**RHS**	Probability
$\mathcal{S} \rightarrow$	B_6	0.4
$\mathcal{S} \rightarrow$	B_8	0.4
$\mathcal{S} \rightarrow$	$\mathsf{B}_8\mathsf{B}_3$	0.1
$\mathcal{S} \rightarrow$	$\mathsf{B}_6\mathsf{B}_6$	0.1
$\mathsf{B}_1 \rightarrow$	1	0.8
$\mathsf{B}_1 \rightarrow$	a	0.2
$\mathsf{B}_3 \rightarrow$	123	1
$\mathsf{B}_6 \rightarrow$	123456	0.7
$\mathsf{B}_6 \rightarrow$	123qwe	0.2
$\mathsf{B}_6 \rightarrow$	dragon	0.1
$\mathsf{B}_8 \rightarrow$	password	0.85
$\mathsf{B}_8 \rightarrow$	p@ssword	0.15

Formally, our context-free grammar (CFG) is defined as $G = (V, \Sigma, \mathcal{S}, R)$, where:

1) $V = \{\mathcal{S}, B_1, B_2, \cdots, B_k$, Capitalize, $L_1, L_2, \cdots, L_6\}$ is a finite set of variables, where k is not larger than the maximum length of a password allowed in the system.
2) Σ =$\mathcal{BS}$=$\{b_1, b_2, \cdots, b_N$, Yes, No$\}$ is a finite set disjoint from V. Note that all 95 printable ASCII codes are in Σ.
3) $\mathcal{S} \in V$ is the start symbol.
4) R is a finite set of rules of the form $A \rightarrow \alpha$, with $A \in V$ and $\alpha \in V \cup \Sigma$ (see Table IV to Table VI).

For instance, password123 $\in \mathcal{T}$ would define the base structure B_{11} and involves no transformation operation, because password123 $\in \mathcal{B}$ and thus password123 can be parsed into just one segment; Password123$\in \mathcal{T}$ would define the base structure B_{11} and involves one capitalization operation, because password123$\in \mathcal{B}$ but Password123$\notin \mathcal{B}$; p@ssw0rd$\in \mathcal{T}$ would define the base structure B_8 and involves one leet operation (i.e., o→0), because p@ssword$\in \mathcal{B}$ but p@ssw0rd$\notin \mathcal{B}$; 123qwe123qwe$\in \mathcal{T}$ defines the base structure $\mathsf{B}_6\mathsf{B}_6$ (i.e., involves one concatenation operation), because the longest prefix is 123qwe$\in \mathcal{B}$. When meeting a password (e.g., tyxdqd123$\in \mathcal{T}$) that can not be parsed (e.g., since there is no item in the trie-tree that is with a prefix tyx), we define this password to be of base structure $\mathsf{B}_8\mathsf{B}_3$, which is similarly parsed by using the traditional PCFG approach [34].

The training phase can automatically derive all the observed base structures, base segments (including basic passwords and other segments) and transformation operations of all passwords in the training set $\mathcal{T}$ and their associated probabilities of occurrence. To see an example of this, please refer to Table IV to Table VI. This builds a probabilistic context-free grammar and then it can be used to measure password strength (i.e., estimate the probabilities of occurrence of a given password) in

TABLE V. PROBABILITIES OF CAPITALIZING THE FIRST LETTER OF A BASE PASSWORD SEGMENT

	First letter capitalized	
	Yes	No
Probability	0.03	0.97

the measuring phase. For instance, from the training phase, we learned that the base structure B_8 occurs with probability 0.4 in the training set, $\mathsf{B}_8 \rightarrow$ `p@ssword` occurs with probability 0.15 and $L_4 \rightarrow$`Yes: o↔0` occurs with probability 0.004. Thus, P(“`p@ssw0rd1`”)= $P(\mathcal{S} \rightarrow \mathsf{B}_8\mathsf{B}_1)\cdot$ $P(\mathsf{B}_8 \rightarrow$ `p@ssword`$)\cdot$ $P(\mathsf{B}_1 \rightarrow$ `1`$)$ $\cdot P$(Capitalize$\rightarrow$`No`) $\cdot P(L_1 \rightarrow$ `No`$)$ $\cdot P(L_2 \rightarrow$ `No`$)$ $\cdot P(L_2 \rightarrow$ `No`$)$ $\cdot P(L_3 \rightarrow$ `Yes`$)$ P(Capitalize $\rightarrow$ `No`) $\cdot P(L_4 \rightarrow$ `No`$)$= $0.1 * 0.15 * 0.8 * 0.97 * 0.994 * 0.995 * 0.995 * 0.004 * 0.97 * 0.997 = 0.00004431$. All the related probabilities are referred to Table IV to VI. This process is shown in Fig. 11.

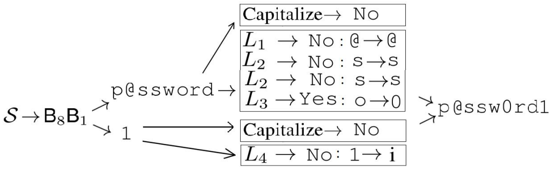


Fig. 11. The derivation of password “p@ssw0rd1” from our fuzzy PCFG

The update phase is used to capture the dynamic changes in the distribution of passwords in a system as time goes on. For instance, after `p@ssw0rd123qwe` is accepted by the system, all the probabilities associated with basic structures B_8 and B_6, terminals `p@ssword` and `123qwe`, and rules $L_3 \rightarrow$ `Yes` shall be updated. This would make our PCFG model more close to the real password distribution, resulting in an adaptive meter. The two PSMs in [33], [34] also provide this feature.

A PCFG is a grammar that has probabilities associated with each rule such that for a specific left-hand side (LHS) variable, all the associated productions add up to 1 [28]. One can see that, our grammar G is indeed a PCFG. The language of G is the set of strings composed of all terminals derived from the start symbol $\mathcal{S}$. When using real-life passwords to instantiate G, over 80% of items in the base structure table are of the form $\mathcal{S} \rightarrow B_m$ (but not $\mathcal{S} \rightarrow B_mB_n$ or more complicated ones). This is in vast contrast with the PCFG-based PSM in [34] where, generally, more than 50% items in its base structure table are of the form $\mathsf{L}_\mathrm{m}\mathsf{D}_\mathrm{n}$ or more complicated ones. In other words, our PCFG is insensitive to a password's traditional structure (i.e., combining segments of words, digits and/or symbol), and thus the notion of “structure” in fuzzyPSM is less clearer than that of PCFG-based PSM. While fuzzy logic aims to imitate the way humans think, our fuzzyPSM aims to imitate the way that humans create passwords for new services. In this light, we call our PCFG a fuzzy one.

Limitations. FuzzyPSM is simple and has several limitations. First, it mainly captures the top-3 popular transformation rules (i.e., concatenation, capitalization and leet), while other rules, such as substring movement and reverse are left as future research. Second, for capitalization, it only considers the capitalization of the first letter of a base password segment. Third, for leet, it only considers the top-6 ones. Still, *fuzzyPSM shows the potential of building a PCFG that more realistically characterizes user behaviors in password creation, and as we will demonstrate in the following section, it in general outperforms all existing PSMs in telling apart weak passwords.*

TABLE VI. PROBABILITIES OF LEET TRANSFORMATION OCCURRENCES OF CERTAIN LETTERS

	$L_1 : a \leftrightarrow @$		$L_2 : s \leftrightarrow \$$		$L_3 : o \leftrightarrow 0$		$L_4 : i \leftrightarrow 1$		$L_5 : e \leftrightarrow 3$		$L_6 : t \leftrightarrow 7$	
	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No
Probability	0.006	0.994	0.005	0.995	0.004	0.996	0.003	0.997	0.002	0.998	0.001	0.999

V. EXPERIMENTAL METHODOLOGIES AND RESULTS

In this section, we first describe our experimental methodologies, including the 11 datasets used and the choices of various training vs. testing scenarios. Then, we use the two rank correlation metrics introduced in Sec. II-C to provide a comparative evaluation of fuzzyPSM with five leading PSMs.

TABLE VII. BASIC INFO ABOUT THE ELEVEN PASSWORD DATASETS

Dataset	Web service	Location	Language	Unique PWs	Total PWs
Tianya	Social forum	China	Chinese	12,898,437	30,901,241
Dodonew	Gaming&E-commerce	China	Chinese	10,135,260	16,258,891
CSDN	Programmer forum	China	Chinese	4,037,605	6,428,277
Zhenai	Dating site	China	Chinese	3,521,764	5,260,229
Weibo	Social forum	China	Chinese	2,828,618	4,730,662
Rockyou	Social forum	USA	English	14,326,970	32,581,870
Battlefield	Game site	USA	English	417,453	542,386
Yahoo	Web portal	USA	English	342,510	442,834
Phpbb	Programmer forum	USA	English	184,341	255,373
Singles.org	Christian dating	USA	English	12,233	16,248
Faithwriters	Christian writing	USA	English	8,346	9,708

A. Dataset descriptions

We employ 11 large-scale real-world password datasets, a total of 97.4 million, to enable a comprehensive evaluation of our meter. The basic information about these datasets are referred to Table VII. Most of them are disclosed by the hacker groups publicly on the Internet. The services range from social forum, gaming, dating, web portal to e-commerce. These passwords are in two most-spoken languages in the world and from two distant continents. As far as we know, our corpus is the largest and the most diversified one ever used for evaluating PSMs (e.g., three datasets in English are used in [33], [34]).

B. Password characteristics

We now have a concrete grasp of what users' real passwords look like. To this end, we investigate the top-10 passwords, character composition and the fraction of overlaps among different password datasets. A number of other characteristics including the top-10 passwords, length distribution and frequency distribution are referred to the full version (see http://t.cn/RG8Ewf3) of this work .

Table VIII shows the character composition information of passwords. Most prominently, a larger fraction of English passwords are composed of only lower-case letters, while a similar fraction of Chinese passwords are composed of only digits. In all, very few passwords (both English and Chinese) incorporate uppercase letters (see the column `[A-Z]`) or symbols (see the column `^[a-zA-Z0-9]+$`).

Fig. 12 shows the fraction of passwords shared between two different services. Generally, the fraction of shared passwords is smaller than 60% considering varied thresholds, and the fraction of shared passwords from different languages (see Tianya vs. Rockyou) is much lower than that of shared passwords from the same language.

Summary. Our results show that passwords from each dataset may be of quite different nature and distributions, which is attributed to a number of confounding factors such as language, service, culture, faith and password policy. This makes it very hard for a PSM to train on the actual distribution (i.e., the “best” training set) in advance. One can only try to make the training set as close to that of the target site as possible.

TABLE VIII. CHARACTER COMPOSITION INFORMATION ABOUT EACH PASSWORD DATASET

Dataset	^[a-z]+$	[a-z]	^[A-Z]+$	**[A-Z]**	^[A-Za-z]+$	[a-zA-Z]	^[0-9]+$	[0-9]	**Symbol only**	**^[a-zA-Z0-9]+$**	^[0-9]+[a-z]+$	^[a-zA-Z]+[0-9]$+	^[0-9]+[a-zA-Z]+$	**^[a-z]+1**$
Tianya	9.91%	34.63%	0.18%	2.96%	10.24%	35.66%	**63.77%**	89.49%	0.03%	98.08%	4.12%	15.73%	4.39%	0.12%
Dodonew	10.30%	66.32%	0.30%	3.67%	10.92%	69.05%	**30.76%**	88.52%	0.02%	98.33%	7.55%	45.74%	7.93%	1.40%
CSDN	11.64%	51.39%	0.47%	4.57%	12.35%	54.33%	**45.01%**	87.10%	0.03%	96.31%	5.88%	28.45%	6.46%	0.24%
Zhenai	6.41%	37.33%	0.24%	3.40%	6.74%	39.54%	**59.52%**	92.87%	0.02%	95.79%	5.24%	21.09%	5.69%	0.08%
Weibo	19.07%	44.77%	0.64%	3.66%	20.55%	46.71%	**53.04%**	78.78%	0.06%	97.79%	2.80%	18.74%	2.91%	1.24%
Rockyou	**41.71%**	80.58%	1.50%	5.94%	**44.07%**	83.89%	15.94%	54.04%	0.02%	96.25%	2.54%	30.18%	2.75%	4.55%
Battlefield	**32.11%**	89.71%	0.29%	9.60%	**34.01%**	90.69%	9.23%	65.49%	0.01%	98.06%	3.05%	39.58%	3.39%	5.08%
Yahoo	**33.09%**	92.83%	0.40%	8.51%	**34.64%**	94.06%	5.89%	64.74%	0.00%	97.15%	5.31%	41.85%	5.64%	4.80%
Phpbb	**50.18%**	86.18%	0.74%	7.70%	**53.07%**	87.83%	12.06%	46.14%	0.03%	98.34%	2.03%	20.94%	2.35%	2.33%
Singles.org	**60.21%**	87.84%	1.92%	8.14%	**65.82%**	90.42%	9.58%	34.06%	0.00%	99.79%	1.77%	19.68%	1.92%	2.73%
Faithwriters	**54.37%**	91.74%	1.16%	8.84%	**58.98%**	93.64%	6.36%	40.88%	0.00%	99.52%	2.37%	25.45%	2.73%	4.13%

Note that the first row is written in regular expressions. For instance, ^[a-z]+$ means passwords that are composed of only lower-case letters; ^[A-Za-z]+$ means passwords composed of only letters; [a-z] means the passwords that include lower-case letters as a substring; ^[0-9]+[a-z]+$ means the passwords composed of digits, followed by lower-case letters.

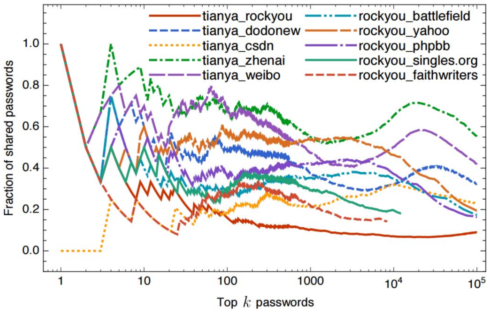


Fig. 12. The fractions of passwords shared between two services.

Fortunately, finding a "good" training set is relatively easy: since the language and service are the two most important factors [37], [45], one can choose a dataset that is of the same language and service with the test set to be the training set, and there are ample (in hundreds) publicly leaked datasets (see [27], [48]) for such choices. We leave for future work to develop a quantized, full-fledged metric to measure the goodness of training set(s). In addition, as new users register, the password distribution will change. This means that no static set of rules can characterize the dynamic changes in password distribution, calling for adaptive meters.

C. Experimental setup

As shown above, password distributions are influenced by a number of factors, among which language is the dominant one. Thus, we divide our training set and testing set into two groups in terms of language. To model user behavior of password reuse, we use Rockyou and Tianya as the base dictionary for fuzzyPSM. For better evaluation, we consider two scenarios: ideal case and real-world case. In the ideal case, the training set for each PSM consists of 1/4 passwords randomly drawn from the testing set, another 1/4 randomly drawn from the testing set are actually measured. As said in IV-A, this eliminates the impacts of improper training set, and the evaluation results only rely on the meter itself.

However, the ideal case is not realistic in practice, because one cannot obtain a training set that is drawn from the same distribution with the testing set. The realistic scenario is that, after the site is set up, the site uses (leaked) passwords from the same language and policy and from a similar service as training passwords to train the PSM, and meanwhile, user-submitted passwords are inserted into the training set and the PSM is dynamically updated. Without loss of generality, this scenario can be modelled as: the PSM is trained on passwords leaked from a similar service and on passwords of the 1/4 testing set, and then it is used to measure all the passwords from the testing set. In addition, we also examine how PSMs perform when trained on passwords from one language but used to measure passwords from another language. All this is summarized in Table IX. Rockyou and Tianya are the weakest in each language group [29], [45], and thus they are chosen as the base dictionaries; Phpbb and Weibo are of moderate strength in each group, and thus they are chosen as the training set for our fuzzyPSM in the real-world case.

TABLE IX. TRAINING AND TESTING SCENARIOS FOR EVERY PSM

Scenario	Base* dictionary	Training set(s)		Testing set, (each set is tested individually)
		Ideal case	Real-world	
English	Rockyou	1/4 testing set	Phpbb, 1/4 testing set	Yahoo,Battlefield,Singles,Faithwriters
Chinese	Tianya	1/4 testing set	Weibo, 1/4 testing set	Dodonew, CSDN, Zhenai
Cross-lan. #1	Rockyou	–	Phpbb, 1/4 Dodonew	Dodonew
Cross-lan. #2	Tianya	–	Weibo, 1/4 Yahoo	Yahoo

* Base dictionary is only used by fuzzyPSM; "Cross-lan." means "Cross-language".

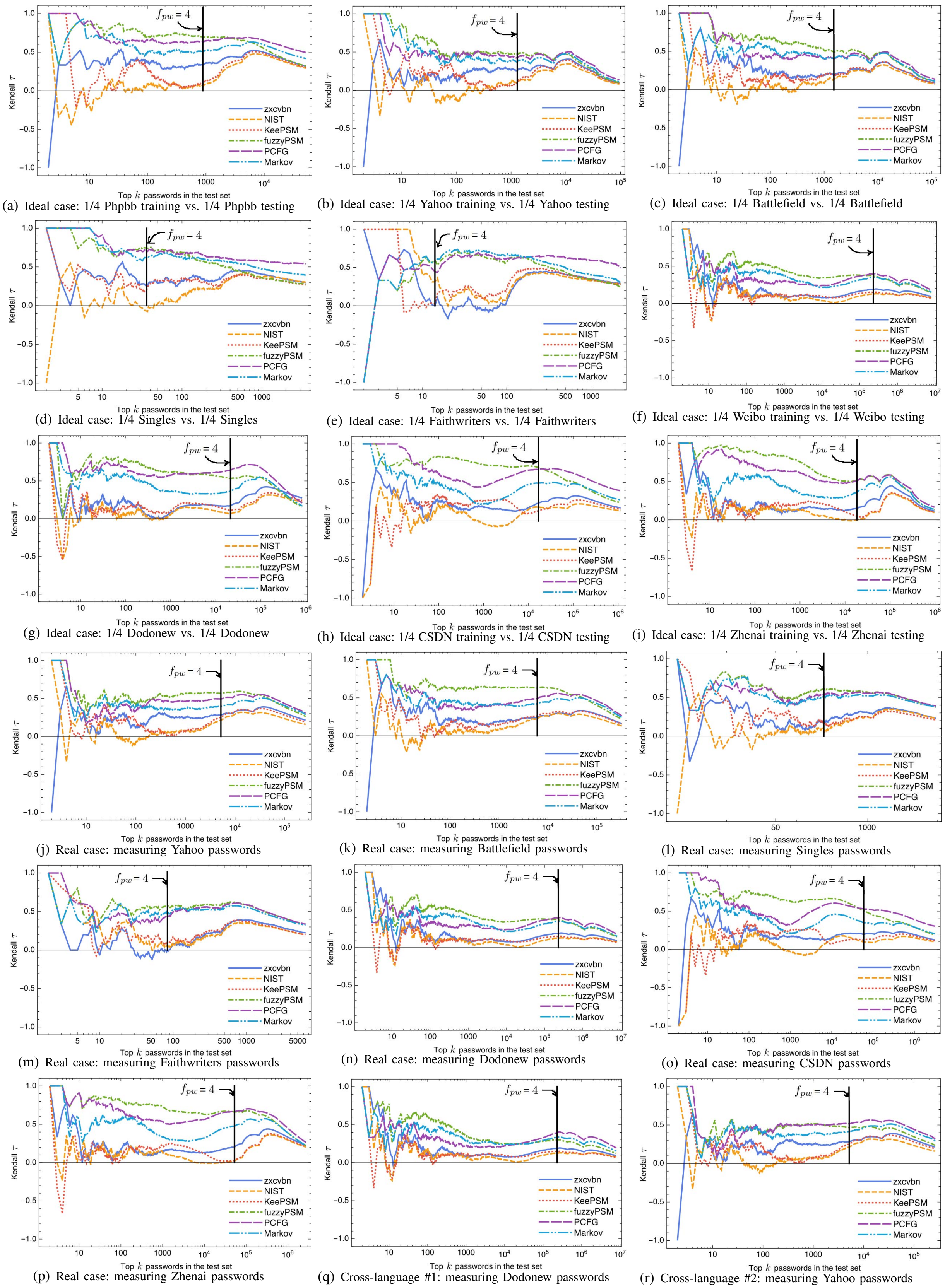
D. Evaluation results

In this section, we compare the performance of the proposed fuzzyPSM with other five leading PSMs, including two from academia (i.e., Markov-based [33] and PCFG-based [34]), two from industry (i.e., Zxcvbn [35] and KeePSM [36]) and one from the standard body (i.e., NIST [16]). The ideal case and real-world case in Table IX would lead to 9 (see Figs. 13(a) to 13(i)) and 7 (see Figs. 13(j) to 13(p)) individual experiments, respectively. We find that, in all our experiments, the Spearman ρ based results show no evident difference from the Kendall τ based results (see Fig. 9(a) vs Fig. 9(b) for example), and thus we only illustrate the latter ones to save space.

In most cases, our fuzzyPSM (represented in the green line in Fig. 13) performs the best when the passwords to be measured are with a frequency no smaller than 4. This is particularly prominent in the real-world cases. As detailed in Sec. II-B, for popular passwords (e.g., with $f_{pw} \geq 4$) one can use the *empirical probability* $f_{pw}/|\mathcal{DS}|$ to approximate its *real probability* p_{pw} with a relative standard error about $1/\sqrt{f_{pw}}$ [41], where f_{pw} is the frequency of pw in the password dataset $\mathcal{DS}$. In other words, only these passwords with $f_{pw} \geq 4$ show its real strength and the ideal meter can *meaningfully* measure them. For passwords with $f_{pw} < 4$, since the ideal meter is ineffective, the benchmark is not reliable, and thus all the comparison results are of little value. In this light, fuzzyPSM performs the best among all examined PSMs.

VI. CONCLUSION

In this paper, we have proposed a simple yet effective password strength meter, fuzzyPSM, to give users reliable feedbacks when they select passwords. Inspired by a user survey on password practice, fuzzyPSM takes the first step towards characterizing realistic user behaviors in password creation. Extensive experiments show that fuzzyPSM outperforms leading PSMs from academia, industry and standard body. Particularly, eleven large-scale password datasets, which consist of 97.4 million real-life passwords and cover various popular services and diversified user bases (e.g., language and culture), are used to establish the practicality of fuzzyPSM.



(a) Ideal case: 1/4 Phpbb training vs. 1/4 Phpbb testing

(b) Ideal case: 1/4 Yahoo training vs. 1/4 Yahoo testing

(c) Ideal case: 1/4 Battlefield vs. 1/4 Battlefield

(d) Ideal case: 1/4 Singles vs. 1/4 Singles

(e) Ideal case: 1/4 Faithwriters vs. 1/4 Faithwriters

(f) Ideal case: 1/4 Weibo training vs. 1/4 Weibo testing

(g) Ideal case: 1/4 Dodonew vs. 1/4 Dodonew

(h) Ideal case: 1/4 CSDN training vs. 1/4 CSDN testing

(i) Ideal case: 1/4 Zhenai training vs. 1/4 Zhenai testing

(j) Real case: measuring Yahoo passwords

(k) Real case: measuring Battlefield passwords

(l) Real case: measuring Singles passwords

(m) Real case: measuring Faithwriters passwords

(n) Real case: measuring Dodonew passwords

(o) Real case: measuring CSDN passwords

(p) Real case: measuring Zhenai passwords

(q) Cross-language #1: measuring Dodonew passwords

(r) Cross-language #2: measuring Yahoo passwords

Fig. 13. Experiment results for the ideal case (see the sub-figures (a)∼(i)) and the real-world case (see the sub-figures (j)∼(p)). Sub-figures (q) an (r) shows the ineffectiveness of cross-language measurement of passwords. The green lines show fuzzyPSM generally performs the best.

Acknowledgment

We thank the anonymous reviewers for their invaluable comments. Ping Wang is the corresponding author. This research was supported by the National Natural Science Foundation of China (NSFC) under Grants Nos. 61472016 and 61501333.

References

[1] J. Bonneau, C. Herley, P. van Oorschot, and F. Stajano, "Passwords and the evolution of imperfect authentication," *Commun. of the ACM*, vol. 58, no. 7, pp. 78–87, 2015.

[2] J. Yan, A. F. Blackwell, R. J. Anderson, and A. Grant, "Password memorability and security: Empirical results." *IEEE Secur. & priv.*, vol. 2, no. 5, pp. 25–31, 2004.

[3] W. Cheswick, "Rethinking passwords," *Commun. of the ACM*, vol. 56, no. 2, pp. 40–44, 2013.

[4] Z. Zhao, G.-J. Ahn, and H. Hu, "Picture gesture authentication: Empirical analysis, automated attacks, and scheme evaluation," *ACM Trans. Inform. Syst. Secur.*, vol. 17, no. 4, pp. 1–37, 2015.

[5] D. Wang and P. Wang, "Two birds with one stone: Two-factor authentication with security beyond conventional bound," IEEE Trans. Depend. Secur. Comput, 2016, in press, available at http://t.cn/RGCaI8f.

[6] J. Bonneau, C. Herley, P. Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proc. IEEE S&P 2012*, pp. 553–567.

[7] C. Herley and P. Van Oorschot, "A research agenda acknowledging the persistence of passwords," *IEEE Secur. & Priv.*, vol. 10, no. 1, pp. 28–36, 2012.

[8] A. Adams and M. A. Sasse, "Users are not the enemy," *Commun. of the ACM*, vol. 42, no. 12, pp. 40–46, 1999.

[9] R. W. Proctor, M.-C. Lien, K.-P. L. Vu, E. E. Schultz, and G. Salvendy, "Improving computer security for authentication of users: Influence of proactive password restrictions," *Behav. Res. Meth. Instr. & Comput.*, vol. 34, no. 2, pp. 163–169, 2002.

[10] S. Egelman, A. Sotirakopoulos, 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*, pp. 2379–2388.

[11] R. Shay, S. Komanduri, P. Kelley, and et al., "Encountering stronger password requirements: user attitudes and behaviors," in *Proc. SOUPS 2010*, pp. 1–20.

[12] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer *et al.*, "How does your password measure up? the effect of strength meters on password creation," in *Proc. USENIX SEC 2012*, pp. 65–80.

[13] B. Ur, F. Noma, J. Bees, S. M. Segreti, R. Shay, L. Bauer, N. Christin, and L. F. Cranor, "I added '!' at the end to make it secure: Observing password creation in the lab," in *Proc. SOUPS 2015*, pp. 123–140.

[14] D. Wang and P. Wang, "The emperor's new password creation policies," in *Proc. ESORICS 2015*, pp. 456–477.

[15] X. Carnavalet and M. Mannan, "A large-scale evaluation of high-impact password strength meters," *ACM Trans. Inform. Syst. Secur.*, vol. 18, no. 1, pp. 1–32, 2015.

[16] W. Burr, D. Dodson, R. Perlner, S. Gupta, and E. Nabbus, "NIST SP800-63-2: Electronic authentication guideline," National Institute of Standards and Technology, Reston, VA, Tech. Rep., Aug. 2013.

[17] M. Weir, S. Aggarwal, M. 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.

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

[19] M. Dell'Amico, P. Michiardi, and Y. Roudier, "Password strength: an empirical analysis," in *Proc. INFOCOM 2010*, pp. 1–9.

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

[21] B. Ur, S. M. Segreti, L. Bauer, and et al., "Measuring real-world accuracies and biases in modeling password guessability," in *Proc. USENIX SEC 2015*, pp. 463–481.

[22] D. Florêncio, C. Herley, and P. C. Van Oorschot, "Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts," in *Proc. USENIX SEC 2014*, pp. 575–590.

[23] F. Zhang, K. Leach, H. Wang, and A. Stavrou, "Trustlogin: Securing password-login on commodity operating systems," in *ASIACCS 2015*.

[24] B. Strahs, C. Yue, and H. Wang, "Secure passwords through enhanced hashing," in *Proc. LISA 2009*, pp. 93–106.

[25] M. Dürmuth and T. Kranz, "On password guessing with GPUs and FPGAs," in *Proc. Password 2014*, pp. 19–38.

[26] C. Allan, *32 million Rockyou passwords stolen*, Dec. 2009, http://www.hardwareheaven.com/news.php?newsid=526.

[27] R. Martin, *Amid Widespread Data Breaches in China*, Dec. 2011, http://www.techinasia.com/alipay-hack/.

[28] 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.

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

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

[31] M. Alsaleh, M. Mannan, and P. Van Oorschot, "Revisiting defenses against large-scale online password guessing attacks," *IEEE Trans. Depend. Secur. Comput.*, vol. 9, no. 1, pp. 128–141, 2012.

[32] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," *Comput. Netw.*, vol. 57, no. 2, pp. 378–403, 2013.

[33] C. Castelluccia, M. Dürmuth, and D. Perito, "Adaptive password-strength meters from markov models," in *Proc. NDSS 2012*, pp. 1–15.

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

[35] D. Wheeler, *zxcvbn: realistic password strength estimation*, April 2012, https://t.co/kLO4dLzqFS.

[36] D. Reichl, *Details on the quality/strength estimations in KeePass*, July 2015, http://keepass.info/help/kb/pw_quality_est.html.

[37] M. Jakobsson and M. Dhiman, "The benefits of understanding passwords," in *Proc. HotSec 2012*, pp. 1–6.

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

[39] R. Graham, *PHPbb Password Analysis*, June 2009, http://www.darkreading.com/risk/phpbb-password-analysis/d/d-id/1130335?

[40] C. Custer, *China's Internet users zoom to 668 million*, July 2015, https://www.techinasia.com/chinas-internet-thirdlargest-country-earth/.

[41] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *Proc. IEEE S&P 2012*, pp. 1–15.

[42] D. Jaeger, H. Graupner, A. Sapegin, F. Cheng, and C. Meinel, "Gathering and analyzing identity leaks for security awareness," in *Proc. Password 2014*, pp. 102–115.

[43] E. Bauman, Y. Lu, and Z. Lin, "Half a century of practice: Who is still storing plaintext passwords?" in *Proc. ISPEC 2015*, pp. 253–267.

[44] D. Florêncio, C. Herley, and P. van Oorschot, "An administrators guide to internet password research," in *Proc. USENIX LISA 2014*, pp. 44–61.

[45] D. Wang, H. Cheng, P. Wang, X. Huang, and C. Chu, "Understanding passwords of chinese users: A survey and empirical analysis," CACR Report, Presented at ChinaCrypt 2015, http://t.cn/RG8RacH.

[46] E. E. Cureton, "The average spearman rank criterion correlation when ties are present," *Psychometrika*, vol. 23, no. 3, pp. 271–272, 1958.

[47] L. M. Adler, "A modification of kendall's tau for the case of arbitrary ties in both rankings," *J. Amer. Statist. Assoc.*, vol. 52, no. 277, pp. 33–35, 1957.

[48] M. Cameron, *All Data Breach Sources*, Feb. 2016, https://breachalarm.com/all-sources.

[49] S. T. Haque, M. Wright, and S. Scielzo, "Hierarchy of users? web passwords: Perceptions, practices and susceptibilities," *Int. J. Hum-Comput. Stud.*, vol. 72, no. 12, pp. 860–874, 2014.

[50] E. Stobert and R. Biddle, "The password life cycle: user behaviour in managing passwords," in *Proc. SOUPS 2014*, pp. 243–255.

[51] K. Murphy, "Dynamic bayesian networks: representation, inference and learning," Ph.D. dissertation, University of California, Berkeley, 2002.