Comparing Password Ranking Algorithms on Real-World Password Datasets

Weining Yang[1], Ninghui Li[1(✉)], Ian M. Molloy[2], Youngja Park[2], and Suresh N. Chari[2]

[1] Purdue University, West Lafayette, IN, USA
{yang469,ninghui}@purdue.edu
[2] IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
{molloyim,young_park,schari}@us.ibm.com

Abstract. Password-based authentication is the most widely used authentication mechanism. One major weakness of password-based authentication is that users generally choose predictable and weak passwords. In this paper, we address the question: How to best check weak passwords? We model different password strength checking methods as Password Ranking Algorithms (PRAs), and introduce two methods for comparing different PRAs: the β-Residual Strength Graph (β-RSG) and the Normalized β-Residual Strength Graph (β-NRSG). In our experiments, we find some password datasets that have been widely used in password research contain many problematic passwords that are not naturally created. We develop techniques to cleanse password datasets by removing these problematic accounts. We then apply the two metrics on cleansed datasets and show that several PRAs, including the dictionary-based PRA, the Markov Models with and without backoff, have similar performances. If the size of PRAs are limited in order to be able to be transmitted over the internet, a hybrid method combining a small dictionary of weak passwords and a Markov model with backoff with a limited size can provide the most accurate strength measurement.

1 Introduction

Password-based authentication is the most widely used authentication mechanism. Despite countless attempts at designing mechanisms to replace it, password-based authentication appears more widely used and firmly entrenched than ever [6,7,21]. One major weakness of password-based authentication is the inherent tension between the security and usability of passwords [4,28]. More precisely, secure passwords tend to be difficult to memorize (i.e., less usable) whereas passwords that are memorable tend to be predictable. Generally individuals side with usability of passwords by choosing predictable and weak passwords [4,18,20,23,30].

To deal with this, the most common approach is to forbid the use of weak passwords, or give warnings for passwords that are "somewhat weak". This approach requires an effective way to identify weak passwords. One way is to use

I. Askoxylakis et al. (Eds.): ESORICS 2016, Part I, LNCS 9878, pp. 69–90, 2016.
DOI: 10.1007/978-3-319-45744-4_4

password composition policies, i.e., requiring passwords to satisfy some syntactical properties, e.g., minimum length and/or categories of characters. An alternative is to use proactive password checkers that are based on a weak password dictionary [5,27,32]. More recently, probabilistic password models, which work by assigning a probability to each password, were introduced [13,26,29,37].

How to best check weak passwords is still an open question. A study in 2014 [14] examined several password meters in use at popular websites and found highly inconsistent strength estimates for the same passwords using different meters. The report did not answer the question of which meter is the best, nor what methods should be used to compare them. Designing an effective password meter requires solving two problems: (1) How to accurately assess the strength of passwords chosen by the users; and (2) How to communicate the strength information to and interact with the users to encourage them to choose strong passwords. These two problems are largely orthogonal. In this paper we focus on solving the first problem.

We model different password strength assessing methods (including composition policies) as Password Ranking Algorithms (PRAs), which assign a rank to every password. One state-of-the-art method for comparing PRAs is the Guess Number Graph (GNG), which plots the number of guesses vs. the percentage of passwords cracked in the test dataset. However, GNG measures only the total density of the uncracked passwords, but not their distribution, which is critical in assessing the effectiveness to defend against guessing attacks after deploying the PRA. To address this limitation of GNG, we propose the β-Residual Strength Graph (β-RSG), which measures the strength of the β most common passwords in the test dataset, after forbidding the weakest passwords identified by a PRA. When a PRA forbids a large number of passwords that users are extremely unlikely to use, it performs poorly under β-RSG. To limit the influence of these passwords, we also propose Normalized β-Residual Strength Graph (β-NRSG), which ignores how passwords that do not appear in the testing dataset are ranked. β-NRSG also has the advantage that we can use it to evaluate blackbox password strength services for which one can query the strength of specific passwords, but cannot obtain all weak passwords.

Surprisingly, we observed that all PRAs perform significantly worse on password datasets from Chinese websites than on datasets from English websites, because some of the most frequent passwords in the testing dataset are not recognized as weak passwords by all the PRAs. Further investigation revealed that these passwords are in all likelihood due to "fake accounts", possibly created by site administrators to artificially boost the number of registered users. The evidences for this include that the user IDs associated with such passwords look suspicious. These suspicious IDs fall in two categories: appending a counter to a fixed prefix; and a large number of fixed-length strings that apparently look random. While these datasets have been used in previous papers, we are the first to report such fake accounts. We developed a data cleansing technique to identify and remove such "fake" accounts in order to obtain a more accurate evaluation.

Our evaluation is based on the cleansed password datasets. We have compared the Probabilistic Context-Free Grammar (PCFG) [37] method, Markov models with and without backoff [13,26,29], blacklists based on training datasets, the Combined method proposed by Ur et al. [34], password composition policies, as well as two versions of *zxcvbn* [38]. We also how GNG, β-RSG, and β-NRSG differ. We found that when one places no limit on the mode size, several methods including the blacklist approach, Markov Models, and the Combined method have similar performance. When one wants to check the strength of passwords on the client side, without sending passwords over the network, the model size must be limited. We found that a blacklist with a limited size still provide the most accurate strength measurement for the most popular passwords. However, only a limited number of passwords are covered.

We then propose a new client-end PRA that uses a hybrid method; it uses a small blacklist to assess the strength of most popular passwords, and evaluate the other passwords based on a limited size Markov model with backoff. We show that the hybrid method inherits the advantages of both methods, and consistently outperform the other client-end PRAs.

The rest of this paper is organized as follows. We discuss related work in Sect. 2. We propose metrics evaluate password ranking algorithms (PRAs) in Sect. 3. The observation of suspicious accounts and the corresponding data cleansing are described in Sect. 4 and the evaluation is reported in Sect. 5. Finally, we conclude in Sect. 6.

2 Related Work

The quality of passwords has traditionally been measured using a combination of standard password cracking tools, such as John the Ripper (JTR) [3], and ad hoc approaches for estimating the information entropy of a set of passwords.

In 1990, Klein et al. [23] proposed the concept of a proactive password checker, which checks the strength of newly created passwords and prevents users from choosing weak passwords. Since then, multiple blacklist-based proactive password checkers were proposed. Spafford et al. [32] and Bergadano et al. [5] developed methods for filtering passwords based on efficiently stored password dictionaries. Manber et al. [27] described an approach that refused not only exact words in dictionaries but also passwords that are a single insertion, deletion, or substitution from a dictionary word. Yan et al. [39] suggested that besides dictionary checking, password checker should consider length and character types of passwords as well.

In terms of entropy estimation, Florencio and Herley [18], Forget et al. [19], and Egelman et al. [16] use the formula where the bit strength of a password is considered to be $\log_2\left(|\Sigma|^{len}\right)$ for alphabet Σ. A more sophisticated approach, known as the NIST guidelines [11], calculates password entropy using several factors, including how many numbers, symbols, and uppercase letters are used and where they appear.

The effect of different password composition policies was studied by Komanduri et al. [25] and Kelley et al. [22] using probability model-based cracking tools. They both suggested that passwords generated under the policy "Password must have at least 16 characters" provides the best security.

In 2010, Weir et al. [36] measured the strength of password creation policies by using large-scale real-world datasets and showed that entropy values would not tell the defender anything about how vulnerable a policy would be to an online password cracking attack. They also showed that many rule-based policies perform poorly for ensuring a desirable level of security. Instead, they suggest measuring the strength of user-chosen passwords by password models and rejecting passwords with high probabilities. Schechter et al. [31] also suggested allowing users to choose any password they want, so long as it is not already too popular with other users.

Probabilistic models of passwords work by assigning a probability to each string. Some models divide a password into several segments, often by grouping consecutive characters of the same category (e.g., lower-case letters, digits, etc.) into one segment, and then generate the probability for each segment independently. Examples include the model in [29], and the Probabilistic Context Free Grammar (PCFG)-based approach developed in [37]. The PCFG approach was later improved in [22,35]. A whole-string model, on the other hand, does not divide a password into segments, e.g., the Markov chain model in [12,13]. Ma et al. [26] showed that Markov chain model with backoff smoothing outperforms PCFG models.

Dell'Amico and Filippone [15] proposed a method to estimate the number of guesses needed to find a password using modern attacks and probabilistic models. Given a probabilistic model, the strength of a passwords is estimated by sampling from the model, i.e., generating random passwords according to the probabilities assigned by the model. This motivates our work to find a way to compare password models, as a better probabilistic model will produce a more accurate estimation.

Recently, Ur et al. [34] compared cracking approaches used by researchers with real-world cracking by professionals. They found that semi-automated cracking by professionals outperforms popular fully automated approaches, but can be approximated by combining multiple approaches and assuming the rank of a password is its highest rank among the approaches examined.

Egelman et al. [16] examined the impact of password meters on password selection and reported that the presence of meters yielded significantly stronger passwords in a laboratory experiment. However, the meters made no observable difference in a field study when creating passwords for unimportant accounts. Ur et al. [33] also showed that scoring passwords stringently results in stronger passwords in general. Komanduri et al. [24] showed that Telepathwords, which makes realtime predictions for the next character that user will type, can help users choosing stronger passwords.

In 2014, de Carné de Carnavalet and Mannan [14] examined several password meters in use at selected popular websites, and revealed how the meters

work. They found gross inconsistencies, with the same password resulting in very different strength across different meters.

In [38], *zxcvbn*, which is an open-source meter designed to run entirely in clients' browser, is proposed. *zxcvbn* decomposes a given password into patterns, and then assigns each pattern an estimated "entropy". The final password entropy is calculated as the sum of its constituent patterns' entropy estimates. The algorithm detects multiple ways of decomposing a password, but keeps only the lowest of all possible summations as an underestimate.

3 How to Compare PRAs

At the core of any password strength meter is a Password Ranking Algorithm (PRA), which is a function that sorts passwords from weak (common) to strong (rare).

Definition 1 (Password Ranking Algorithm (PRA)). *Let $\mathcal{P}$ denote the set of all allowed passwords. A Password Ranking Algorithm $r : \mathcal{P} \rightarrow \mathsf{Rnk}$ is a function that maps each password to a ranking in* Rnk*, where* $\mathsf{Rnk} = \{1, \ldots, |\mathcal{P}|\} \cup \{\infty\}$.

Intuitively, a password with rank 1 means that it is considered to be one of the weakest password(s); and a password with rank ∞ means that it is considered to be strong enough to not need a ranking. The above definition accommodates PRAs that rank only a subset of all passwords as well as PRAs that rank some passwords to be of equal strength. A password composition policy can be modeled as a PRA that assigns a rank of 1 to passwords that do not satisfy the policy, and ∞ otherwise. Probabilistic password models that assign a probability to each password can be converted into a PRA by sorting, in decreasing order, the passwords based on their probabilities in the model. Arguably, this captures the essential information for determining the strengths of passwords, since both cracking passwords and choosing which passwords to forbid should be done based on the ranking.

3.1 Guess Number Graph (GNG)

The state-of-the-art method for comparing PRAs is the Guess Number Graph (GNG), which plots the number of guesses vs. the percentage of passwords cracked in the dataset. A point (x, y) on a curve means that y percent of passwords are included in the first x guesses. When evaluating PRAs for their effectiveness in cracking passwords, GNG is an ideal choice. For the same x value, a PRA that has a higher y value is better. However, one limitation of GNG is that it does not convey information regarding the distribution of uncracked passwords. For example, suppose that two PRAs r_1 and r_2 both cover 40 % of passwords after making 10^6 guesses. Under r_1 there remain uncovered 5 passwords each appearing 200 times and a large number of passwords that appear just once. And under r_2 there remain 500 passwords each appearing 2 times together with

a similarly large number of passwords that appear just once. In this case, if we decide to forbid the first 10^6 passwords that are considered weak and an adversary is limited to 5 guess attempts per account (e.g., because of rate limiting), the adversary can successfully break into 1,000 accounts based on r_1, but only 10 accounts based on r_2. Obviously, r_1 is worse than r_2, even though they look the same under the GNG. Therefore, GNG is not appropriate for the effectiveness of using a PRA for identifying and forbidding the usage of weak passwords, especially since the primary objective of checking password strength is to defend against online guessing attacks, as offline attacks are best defended against by improving site security and by using salted, slow cryptographic hash functions when storing password hashes.

3.2 The β-Residual Strength Graph (β-RSG)

To deal with the limitation of GNG, we propose to use the β-Residual Strength Graph. Each PRA r corresponds to a curve, such that a point (x, y) on the curve means that after forbidding what r considers to be the x weakest passwords, the strength of the remaining passwords is y. For the choice of y, we use the effective key-length metric corresponding to the β-success-rate, proposed by Boztaş [8] and Bonneau [6], to measure the strength of the probabilities of the remaining passwords, which we call the residual distribution.

More specifically, given a password dataset D, we use $p_D(w)$ to denote a password w's frequency in D, i.e., $p_D(w) = \frac{\text{number of times } w \text{ occurs in } D}{|D|}$. Given a PRA r and a number x, let w_i be the ith most frequent password in D that is not among the x weakest passwords according to r. Then the β-Residual Strength is computed as:

$$y = \lg\left(\frac{\beta}{\sum_{i=1}^{\beta} p_D(w_i)}\right),$$

Intuitively, β-RSG provides a measure of the strength of the remaining weakest passwords after a certain number of weak passwords according to r are forbidden. It translates the total frequencies of the β unremoved weakest passwords into a bit-based security metric, which can be viewed as finding the entropy of a uniform distribution where the probability of each element equals that of the average of these β passwords.

We need to choose appropriate values for β. In [6], $\tilde{\lambda}_{10}$ is used, which corresponds to an online attack setting where 10 guesses are allowed, which was recommended by usability studies [9]. We adapt the setting.

3.3 The Normalized β-Residual Strength Graph (β-NRSG)

Password composition policies (such as the ones that require mixing letters with digits and special symbols), when viewed as PRAs, tend to perform poorly under the RSG, because they rule out a large number of passwords, e.g., all passwords that consist of only letters. This demonstrates that one weakness of password

composition policies is that they prevent some strong passwords (such as unpredictable passwords consisting of only letters) from being used. However, one may argue that this is not completely fair to them. The cost of forbidding a strong (i.e., rarely used) password is that users who naturally want to use such a password cannot do so, and have to choose a different password, which they may have more trouble remembering. However, if users are extremely unlikely to choose the password anyway, then there is very little cost to forbid it.

We thus propose a variation of RSG, which "normalizes" a RSG curve by considering only passwords that actually appear in the testing dataset D. More specifically, a point (x, y) on the curve for a PRA r means that after choosing a threshold such that x passwords that appear in D are forbidden, the residual strength is y. We call this the Normalized β-Residual Strength Graph (β-NRSG). A NRSG curve can be obtained from a corresponding RSG curve by shrinking the x axis; however, different PRAs may have different shrinking effects, depending on how many passwords that are considered weak by the PRAs do not appear in the testing dataset. Under β-NRSG, PRAs are not penalized for rejecting passwords that do not appear in the testing dataset. A PRA would perform well if it considers the weak (i.e., frequent) passwords in the dataset to be weaker than the passwords that appear very few times in it. β-NRSG also has the advantage that we can use it to evaluate blackbox password strength services for which one can query the strength of specific passwords, but cannot obtain all weak passwords. We suggest using both RSGs and NRSGs when comparing PRAs.

3.4 Client Versus Server PRAs

A PRA can be deployed either at the server end, where a password is sent to a server and has its strength checked, or at the client end, where the strength checking is written in JavaScript and executed in the client side inside a browser. PRAs deployed at the server end are less limited by the size of the model. On the other hand, deploying PRAs on the client side increases confidence in using them, especially when password strength checking tools are provided by a third party. Thus it is also of interest to compare the PRAs that have a relatively small model size, and therefore can be deployed at the client end. We say a PRA is a Client-end PRA if the model size is less than 1 MB, and a Server-end PRA otherwise.

Table 1. Server-end PRAs and Client-end PRAs. X_c means reduced-size version of model X in order to be deployed at the client side.

Server-end	Markov Model [13], Markov Model with backoff [26], Probabilistic Context-free Grammar [37], Google API, Blacklist, Combined [34]
Client-end	$zxcvbn_1$ [38], $zxcvbn_2$ [38], Blacklist$_c$, Markov Model$_c$, Markov Model with backoff$_c$, Hybrid

3.5 PRAs We Consider

The PRAs that are considered in this paper are listed in Table 1. In Client-end PRAs, the size of $zxcvbn_1$, $zxcvbn_2$ are 698 KB and 821 KB correspondingly. For password models whose model sizes are adjustable, we make the model size to be approximately 800 KB to have a fair comparison.

PCFG. In the PCFG approach [37], one divides a password into several segments by grouping consecutive characters of the same category (e.g., letters, digits, special symbols) into one segment. Each password thus follows a pattern, for example, L_7D_3 denotes a password consisting of a sequence of 7 letters followed by 3 digits. The distribution of different patterns as well as the distribution of digits and symbols are learned from a training dataset. PCFG chooses words to instantiate segments consisting of letters from a dictionary where all words in the dictionary are assumed to be of the same probability. The probability of a password is calculated by multiplying the probability of the pattern by the probabilities of the particular ways the segments are instantiated.

Markov Model. N-gram models, i.e., Markov chains, have been applied to passwords [13]. A Markov chain of order d, where d is a positive integer, is a process that satisfies

$$P(x_i|x_{i-1}, x_{i-2}, \ldots, x_1) = P(x_i|x_{i-1}, \ldots, x_{i-d})$$

where d is finite and $x_1, x_2, x_3, \ldots$ is a sequence of random variables. A Markov chain with order d corresponds to an n-gram model with $n = d + 1$.

We evaluate 5-gram Markov Model (MC_5), as recommended in [26], within Server-end PRAs setting. In order to fit the Markov Model into a Client-end PRA, if we store the frequency of each sequence in a trie structure, the leaf level contains 95^n nodes, where 95 is the total number of printable characters. To limit the size of Markov model to be no larger than 1 MB, n should be less than 4. We use 3-order Markov Model MC_3 in our evaluation.

Markov Model with Backoff. Ma et al. [26] proposed to use the Markov Model with backoff to model passwords. The intuition is that if a history appears frequently, then we would want to use that to estimate the probability of the next character. In this model, one chooses a threshold and stores all substrings whose counts are above the threshold, and use the frequency of these substrings to compute the probability. Therefore, the model size of a Markov Model with backoff depends on the frequency threshold selected. In this paper, we consider two sizes of Markov Model with backoff by varying frequency threshold. We first pick a relatively small threshold 25 (MCB_{25}), as suggested in [26], to construct a Server-end PRA.

For Client-end PRAs, similar to the Markov model, we record the model in a trie structure, where each node contains a character and the corresponding count of the sequence starting from the root node to the current node. We measure the size of data after serializing the trie into JSON format. Table 3 shows the size of the models trained on *Rockyou* and *Duduniu* dataset with different

frequency thresholds. The size of the Markov Models with backoff when trained on *Duduniu* dataset is significantly smaller than that of models trained on the *Rockyou* dataset. This is primarily due to the difference in character distribution between English and Chinese users. English users are more likely to use letters while Chinese users are more likely to use digits. As a result, the most frequent sequences in *Rockyou* are mainly constructed by letters while those in *Duduniu* are mainly constructed by digits. The difference in the size of the models comes from the different search space in letters and digits. In order to approximate the size of the model to that of *zxcvbn*, we choose MCB_{1500} for English datasets and MCB_{500} for Chinese datasets.

Dictionary-Based Blacklist. Dictionary-based blacklists for filtering weak passwords have been studied for decades, e.g., [5,27,32]. Some popular websites, such as Pinterest and Twitter, embed small weak password dictionaries, consisting of 13 and 401 passwords respectively, on their registration pages. We use a training dataset to generate the blacklist dictionary. The order of the passwords follows the frequency of passwords in the training dataset in a reversed order. Assuming each password contains 8 characters on average, a dictionary with 100,000 passwords is approximately 900KB. Such blacklist (Blacklist_c) is used in Client-end PRAs settings.

Combined Method. Ur et al. [34] proposed Min_{auto} metric, which is the minimum guess number for a given password across multiple automated cracking approaches. We implement a password generator which outputs passwords in the order of their corresponding Min_{auto}. Passwords with smaller Min_{auto} are generated earlier. In the Combined PRA, the rank of a password is the order of the passwords generated. In this paper, Min_{auto} is calculated by combining 4 well-studied approaches: Blacklist, PCFG, Markov, and Markov with backoff.

Google Password Strength API. Google measures the strength of passwords by assigning an integer score ranging from 1 to 4 when registering on their website. We found that the score is queried via an AJAX call and the API is publicly available[1]. We use this service to assess the strength of passwords. We are not able to generate passwords and get the exact ranking as the underlying algorithm has not been revealed.

Zxcvbn Version 1. Zxcvbn is an open-source password strength meter developed by Wheeler [38]. It decomposes a given password into chunks, and then assigns each chunk an estimated "entropy". The entropy of each chunk is estimated depending on the pattern of the chunk. The candidate patterns are "dictionary", "sequence", "spatial", "year", "date", "repeat" and "bruteforce". For example, if a chunk is within the pattern "dictionary", the entropy is estimated as the log of the rank of word in the dictionary. Additional entropy is added if uppercase letters are used or some letters are converted into digits or sequences (e.g. a⇒@). There are 5 embedded frequency-ordered dictionaries:

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

7140 passwords from the Top 10000 password dictionary; and three dictionaries for common names from the 2000 US Census. After chunking, a password's entropy is calculated as the sum of its constituent chunks' entropy estimates.

$$\mathsf{entropy}(pwd) = \sum \mathsf{entropy}(chunk_i)$$

A password may be divided into chunks in different ways, Zxcvbn finds the way that yields the minimal entropy and uses that.

Zxcvbn Version 2. In October 2015, a new version of *zxcvbn* was published. Zxcvbn_2 also divides a password into chunks, and computes a password's strength as the "minimal guess" of it under any way of dividing it into chunks. A password's "guess" after being divided into chunks under a specific way is:

$$l! \times \prod_{i=1}^{l} (chunk_i.guesses) + 10000^{l-1}$$

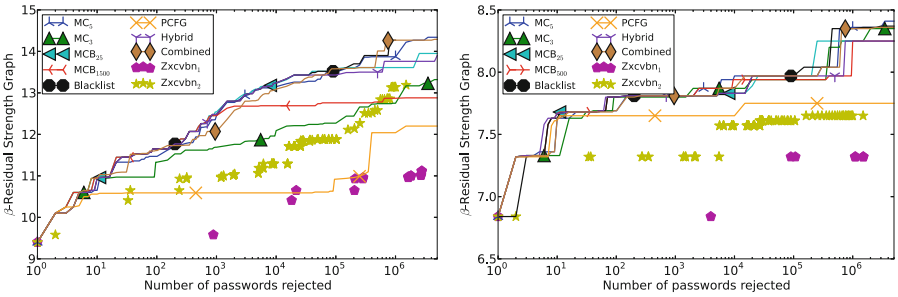
where l is the number of the chunks. The factorial term is the number of ways to order l patterns. The $10000^{(l-1)}$ term captures the intuition that a password that has more chunks are considered stronger. Another change in the new version is that if a password is decomposed into multiple chunks, the estimated guess number for each chunk is the larger one between the chunks' original estimated guess number and a min_guess_number, which is 10 if the chunk contains only one character or 50 otherwise. While these changes are heuristic, our experimental results show these changes cause significant improvements under our methods of comparison.

Hybrid Method. Observing the promising performance of dictionary methods and the limited number of passwords covered (see Sect. 5.2 for details), we propose a hybrid PRA which combines a blacklist PRA with a backoff model. In the hybrid PRA, we reject passwords belonging to a blacklist dictionary or with low scores using the backoff model. To make the size of the PRAs consistent, we further limit the size for both dictionary and backoff model. We chose to use a dictionary containing 30 000 words, which takes less than 300KB. In order to keep the total size of the model consistent, we used MCB_{2000} and MCB_{1000} for English datasets and Chinese datasets, respectively.

4 Data Cleansing

Poor Performance of PRAs on Chinese Datasets. In our evaluation comparing PRAs, we observe that almost all PRAs perform poorly on some Chinese dataset.

Figure 1 shows the results of an β-Residual Strength Graph(β-RSG) evaluation on *Xato* (an English dataset) and *178* (a Chinese dataset). A point (x, y) on a curve means if we want to reject top x passwords from a PRA, the residual strength is y. It is clear that the residual strength for *178* is much lower than that



(a) *Xato*, $\beta = 10$ (b) *178*, $\beta = 10$

Fig. 1. β-Residual Strength Graph(β-RSG) on original *Xato* and *178* datasets. A point (x, y) on a curve means if we want to reject top x passwords from a PRA, the strength of the remaining passwords is y.

of *Xato*. In *178*, even if 1 million passwords are rejected, the residual strength is around or lower than 8 for all PRAs we examined, which means the average of the remaining top 10 passwords's probability is as high as $\frac{1}{2^8} \approx 0.39\,\%$. We found that 12 out of the top 20 passwords in *178* were not among the first million weakest passwords for any PRA. This led us to investigate why this occurs.

Evidences of Suspicious IDs. We found that the dataset contains a lot of suspicious account IDs which mostly fall in to two patterns: (1) *Counter*: a common prefix appended by a counter; (2) *Random*: a random string with a fixed length. Table 2 lists some suspicious accounts sampled from the *178* dataset, which we believe were created either by a single user in order to benefit from the bonus for new accounts, or by the system administrator, in order to artificially boost the number users on the sites. Either way, such passwords are not representative of actual password choices and should be removed.

Table 2. Examples of IDs in *178* Dataset.

Password	*Counter* IDs (sampled)	*Random* Ids (sampled)
`zz12369`	badu1; badu2; ...; badu50	vetfg34t; gf8hgoid; vkjjhb49; 5t893yt8; 9y4tjreo; 09rtoivj; kdznjvhb
`qiulaobai`	qiujie0001; qiujie0002; ...; qiujie0345	j3s1b901; ul2c6shx; a3bft0b8; wzjcxytp; 7fmjwzg2; 0ypvjqvo
`123456`	1180ma1; 1180ma2; ...; 1180ma49	x2e03w5suedtu; 7kjwddqujornc; inrrgjhm2dh8r; 3u2lnalg91u9i;

Suspicious Account Detection. We detect and remove suspicious passwords (accounts) using the user IDs and email addresses. *Yahoo* and *Duduniu* datasets only have email address available. We first remove the email provider, i.e., the postfix starting from @, and then, treat the prefix of email addresses as account IDs.

Table 3. Model size of Markov Model with backoff using different frequency threshold.

Train	Frequency Threshold					
	25	200	500	1000	1500	2000
RockYou	18 M	3.4 M	1.7 M	1 M	712 K	556 K
Duduniu	7.8 M	1.5 M	604 K	368 K	268 K	200 K

Table 4. Number of accounts removed.

Dataset	*Yahoo*	*Xato*	*Duduniu*	*CSDN*	*178*
Removed	232	9577	9796	69317	1639868
Total	434131	9148094	7304316	6367411	8434340

Rockyou and *Phpbb* datasets are excluded in the following analysis, as we do not have access to user IDs/emails.

We identify *Counter* IDs utilizing Density-based Spatial Clustering of Applications with Noise (DBSCAN) [17]. DBSCAN is a density-based clustering algorithm. It groups together points that are closely packed together (a brief overview of DBSCAN is provided in Appendix A). In our case, each ID is viewed as a point, and the distance between two IDs are measured by the Levenshtein distance, which measures the minimum number of single-character edits. Given a password, we first extract all the corresponding IDs in the dataset, and then generate groups of IDs, where the IDs in the same group share a common prefix with length at least 3. The grouping is introduced to reduce the number of points to be clustered, as calculating pairwise distance of a large number of data points is slow. Next, we apply DBSCAN with $\epsilon = 1$ and $minPts = 5$ to each group. Finally, we label all IDs in clusters with size at least 5 as suspicious.

Random. IDs are identified based on probabilities of IDs, which are calculated utilizing a Markov Model. Intuitively, *Random* IDs are ids whose probabilities are "unreasonably small". Observing that *Random* IDs are generally with the same length and the probabilities of IDs can be approximated by lognormal distribution (see Appendix B), we perform "fake" account removal for IDs with the same length based on $-\log p$, where p is probabilities of IDs. Note that in a normal distribution, nearly all values are within three standard deviations of the mean (three-sigma rule of thumb), we therefore, believe $\mu + 3\sigma$ is a reasonable upper-bound for "real" IDs, where μ and σ are mean and standard deviation of $-\log p$, respectively.

In addition, if most of the IDs corresponding to a high-frequency password P in dataset D are detected as suspicious, and P does not appear in password datasets other than D, we remove all accounts associated with the P.

Table 5. Top 5 Passwords with Most Accounts Removed. $pwd_{r/o}$ means the original count of *pwd* in the dataset is o, and r accounts are removed.

Rank	*Yahoo*	*Xato*	*Duduniu*	*Csdn*	178
1	$\text{1a1a1a1b}_{131/131}$	$\text{klaster}_{1705/1705}$	$\text{aaaaaa}_{3103/10838}$	$\text{dearbook}_{44636/44636}$	$\text{qiulaobai}_{57963/57963}$
2	$\text{welcome}_{101/437}$	$\text{iwantu}_{885/885}$	$\text{111111}_{1203/21763}$	$\text{xiazhili}_{3649/3649}$	$\text{wmsxie123}_{48258/49162}$
3	-	$\text{1232323q}_{450/450}$	$\text{123456}_{1076/93259}$	$\text{12345678}_{2222/212743}$	$\text{123456}_{47536/261692}$
4	-	$\text{galore}_{393/393}$	$\text{9958123}_{461/3981}$	$\text{123456789}_{1482/234997}$	$\text{w2w2w2}_{35762/35762}$
5	-	$\text{wrinkle1}_{243/243}$	$\text{a5633168}_{457/457}$	$\text{11111111}_{1301/76340}$	$\text{wolf8637}_{31909/31909}$

Results of Cleansing. Table 4 lists the number of suspicious accounts removed. In general, the suspicious accounts count for a small portion in English and *Duduniu* datasets. However, the number of suspicious accounts detected in *CSDN* and *178* datasets are significantly larger. In *178* dataset, about one fifth accounts are suspicious. Table 5 lists the top 5 passwords with most accounts removed in each dataset. Despite the accounts correspond to uncommon passwords, a significant number of accounts with popular passwords, such as `123456`, are removed as well. Evidences suggest that some datasets contain many waves of creation of suspicious accounts, some using common passwords such as `123456`, as illustrated in Table 2.

5 Experimental Results

5.1 Experimental Datasets and Settings

We evaluate PRAs on seven real-world password datasets, including four datasets from English users, *Rockyou* [1], *Phpbb* [1], *Yahoo* [1], and *Xato* [10], and three datasets from Chinese users, CSDN [2], *Duduniu*, and *178*.

Some PRAs require a training dataset for preprocessing. For English passwords, we train on *Rockyou* and evaluate on (1) *Yahoo* + *Phpbb*; (2) *Xato*, as *Rockyou* is the largest password dataset available. We combine *Yahoo* and *Phpbb* datasets because the size of them are relatively small. For Chinese passwords, the evaluation was conducted on any pair of datasets. For each pair, we trained PRAs on one dataset and tested on the other. Because of the page limit, we only present results of using *Duduniu* as the training dataset.

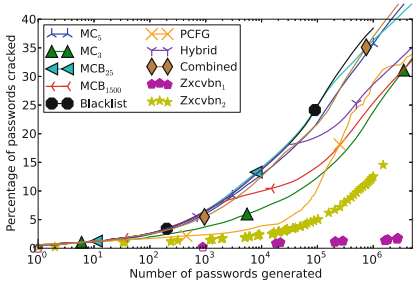
Probabilistic Password Models. For all probabilistic password models we evaluate, we generate 10^8 passwords following the descending order of probabilities. The order of the password generated is essentially the ranking of the password in the corresponding PRA.

Blacklist PRAs. We directly use the training dataset as blacklist. Namely, in the PRA, the ranking of a password is the order of its frequency in the training dataset. We vary the size of blacklist by removing the lowest-rank passwords in order to adjust the number of passwords rejected.

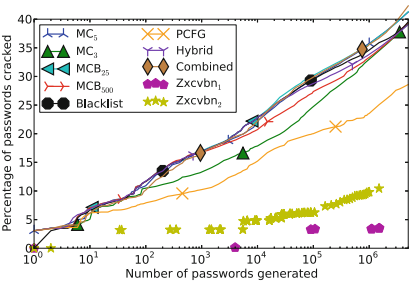
Zxcvbn. *Zxcvbn* was designed to evaluate entropy for passwords from English speaking users only. When applied to Chinese datasets, we modify it by adding a new dictionary of Chinese words. In addition, we implemented a password generator which generate passwords in the order of entropy measured by the model. The implementation details are in Appendix C.

5.2 Experimental Results

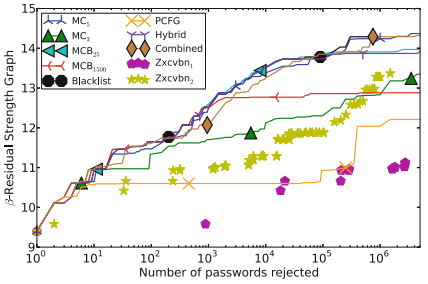
Figure 2 illustrates the Guess Number Graph (GNG), the β-Residual Strength Graph (β-RSG), and the Normalized β-Residual Strength Graph (β-NRSG) evaluated on *Xato* and *178* datasets. The corresponding training datasets are *Rockyou* and *Duduniu*, respectively. The evaluation on the other datasets leads to similar results.



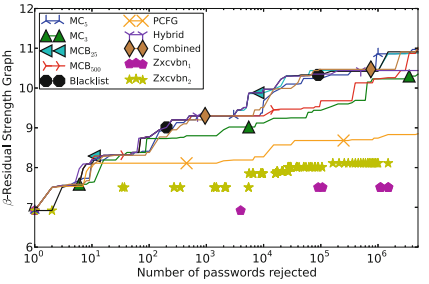
(a) GNG, *Xato*



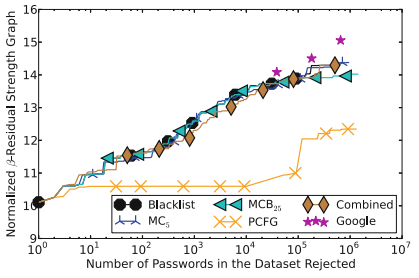
(b) GNG, *178*



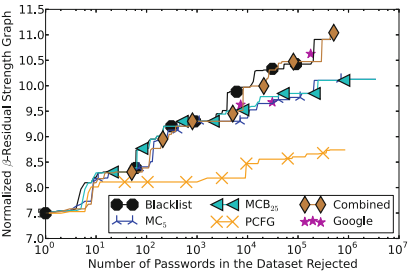
(c) β-RSG, *Xato*, $\beta = 10$



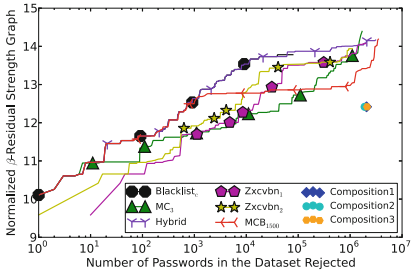
(d) β-RSG, *178*, $\beta = 10$



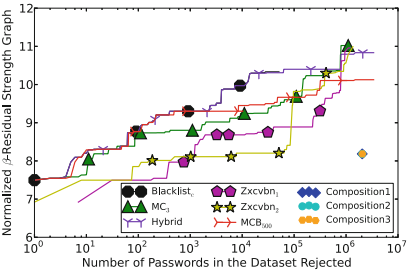
(e) β-NRSG, *Xato*, Server-end, $\beta = 10$



(f) β-NRSG, *178*, Server-end, $\beta = 10$



(g) β-NRSG, *Xato*, Client-end, $\beta = 10$



(h) β-NRSG, *178*, Client-end, $\beta = 10$

Fig. 2. The Guess Number Graph (GNG), the β-Residual Strength Graph (β-RSG), and the Normalized β-Residual Strength Graph (β-NRSG) evaluated on *Xato* and *178* datasets.

Figure 2(a) and (b) show the evaluation of the Guess Number Graph (GNG). Both Client-end and Server-end PRAs, except Google's password strength assessment from which we are not able to generate passwords, are measured. We do not plot the Blacklist PRA with limited size, as it overlaps with the regular Blacklist PRA. We plot scatter points for *zxcvbn* to avoid ambiguity, since it generates multiple passwords with the same entropy. A point (x, y) on a curve means that y percent of passwords in the test dataset are included in the first x guesses.

Figure 2(c) and (d) illustrate the β-Residual Strength Graph (β-RSG) for $\beta = 10$. In the evaluation, we vary the number of passwords rejected x in PRAs (i.e., passwords ranked as top x are not allowed). In the figures, a point (x, y) on a curve means if we want to reject top x passwords from a PRA, the residual strength is y. For a fixed x, a larger y indicates smaller portion of accounts will be compromised within β guesses after rejecting x passwords. Comparing Fig. 1(b) and Fig. 2(d), we can observe that the performance of PRAs on cleansed data significantly boost, which confirm the need of data cleansing.

The Normalized β-Residual Strength Graphs (β-NRSG) for Server-end PRAs are illustrated in Fig. 2(e) and (f), and the Client-end PRAs' evaluation is shown in Fig. 2(g) and (h). In addition to PRAs compared in GNG and β-RSG, we evaluate the effect of composition policies and Google's password strength API as well. Three commonly used composition rules are examined. Composition rule 1 is adapted by Ebay.com, which ask for at least two types of characters from digits, symbols and letter. Composition rule 2 is adapted by Live.com, which also ask for two types of characters, but it further split letters into uppercase and lowercase letters. Composition rule 3 is adapted by most of the online banking sites (e.g. BOA). At least one digit and one letter are required.

Server-end PRAs. In general, Server-end PRAs (Blacklist, MC_5, MCB_{25}, Combined) outperform Client-end PRAs (Hybrid, MC_3, MCB_{1500}/MCB_{500}), which confirms that a PRA's accuracy grows with the size of its model, and Server-end PRAs are recommended for websites where security is one of the most important factors, e.g., online banks.

The Google password strength API, which is only evaluated in β-NRSG (Fig. 2(e) and (f)) is the top performer on both English and Chinese datasets. The three points from left to right in each graph illustrate the effect of forbidding passwords whose score is no larger than 1, 2, and 3, respectively. In practice, all passwords with score 1 are forbidden. The high residual strength indicates that most of the high-frequency passwords are successfully identified.

For the other Server-end PRAs, the three metrics (Fig. 2(a)-(f)) all suggest that several PRAs including the Blacklist PRA, the Markov Model with backoff with a frequency threshold of 25 (MCB_{25}) [26], the 5-order Markov Model [26], and the Combined method [34] have similar performance, and they are almost always on the top of the graphs, which is consistent with the results in the previous works [15,26,34].

Client-end PRAs. From Fig. 2(g) and (h), it is clear that composition rules do not help prevent weak passwords, as the corresponding points are far below the

other curves. In addition, the composition rules generally reject more than one tenth of passwords in the datasets, which might lead to difficulty and confusion in password generation, and is not appropriate.

Among the other Client-end PRAs, the Blacklist PRA outperform the others when the number of passwords rejected is small. However, because of the limited size, the small blacklist can only cover a small proportion of passwords (less than 10,000) in the testing dataset. The reduced-size Markov models (MC_3 and MCB_c) perform significantly worse than the corresponding Server-end models (MC_5 and MCB_{25}), especially when the number of passwords rejected is relatively large. The low order Markov models cannot capture most of the features in the real passwords distribution and the strength measurement is not accurate. MCB_c performs similar to the Blacklist PRA when x is small, as the frequencies of the most popular patterns are high enough to be preserved, with the cost of losing most of the other precise information. As a result, the performance of MC_3 is better than MCB_c with the growth of x.

A noticeable improvement of $zxcvbn_2$ over $zxcvbn_1$ can be observed in all the three metrics (Fig. 2(a)-(d), (g), and (h)). The figures also suggest that *zxcvbn* is not optimized for passwords created by non-English speaking users, as the performance of the PRAs significantly drops in the evaluation on Chinese datasets.

The Hybrid Method. Observing the promising performance of Blacklist methods and the limited number passwords covered in the testing dataset, we propose a hybrid PRA which combines a blacklist PRA with a backoff model. In the Hybrid PRA, we first reject passwords based on the order in the Blacklist, and apply the backoff model after the Blacklist is exhausted. To make the size of the PRA consistent, we further limit the size for both the Blacklist and Markov Model with backoff. We set the frequency threshold to 2000 for the English password datasets and 1000 for the Chinese password datasets (see Table 3 for model sizes). We further reduce the size of the Blacklist to 30,000 words, resulting in a dictionary smaller than 300 KB. The total size of the hybrid model is less than 800KB. The figures (Fig. 2(a)-(d), (g), and (h)) show that the hybrid method inherits the advantage of Blacklist PRA and Markov Model with backoff. Hybrid method can accurately reject weak passwords, and can provide a relatively accurate strength assessment for any passwords. As a result, it is almost always on the top of all client-end PRAs, and is even comparable with Server-end PRAs in β-RSG and β-NRSG measurements.

Differences Among the Three Metrics. Table 6 lists the y values in GNG and β-RSG when $x = 10^4$ and $x = 10^6$. From the table, we can observe that although the percentage of passwords cracked by PRAs significantly increase from when rejecting ten thousand passwords to when rejecting one million passwords, the difference between y values in β-RSG is limited, especially for the top-performing PRAs, such as the blacklist method. The different behavior between GNG and β-RSG indicates that the percentage of passwords cracked, which is shown in GNG, cannot infer the residual strength, which is the observation from β-RSG. A high coverage and a low coverage in password cracking might result in similar residual strength, as the most frequent remaining passwords might

Table 6. y values of GNG and β-RSG when $x = 10^4$ and $x = 10^6$. *Y+P* stands for *Yahoo + Phpbb*. $\beta = 10$

	English Datasets								Chinese Datasets							
	GNG				RSG				GNG				RSG			
Dataset	*Y+P*		*Xato*		*Y+P*		*Xato*		*CSDN*		*178*		*CSDN*		*178*	
x	10 K	1 M	10 K	1 M	10 K	1 M	10 K	1 M	10 K	1 M	10 K	1 M	10 K	1 M	10vK	1 M
MC_5	14 %	34 %	13 %	36 %	12.7	13.1	13.4	14.2	16 %	26 %	22 %	36 %	10.2	10.3	9.7	10.9
MC_3	7.3 %	21 %	6.9 %	24 %	11.5	12.4	12.1	12.8	13 %	23 %	18 %	33 %	9.7	9.9	9.1	10.2
MCB_{25}	16 %	35 %	14 %	36 %	12.8	13.2	13.5	13.9	17 %	27 %	23 %	36 %	10.3	10.4	9.9	10.4
MCB_c	11 %	22 %	10.0 %	25 %	12.6	12.7	12.8	12.9	16 %	25 %	21 %	33 %	10.1	10.3	9.3	10.2
zxcvbn	0.7 %	1.4 %	0.5 %	1.3 %	10.1	10.8	10.0	11.0	0.1 %	3.5 %	0.3 %	3.4 %	6.6	7.1	7.0	7.5
$zxcvbn_{v2}$	2.5 %	13 %	2.4 %	13 %	11.2	12.8	11.3	13.3	3.5 %	11 %	4.8 %	9.8 %	7.1	8.9	7.8	8.1
Blacklist	16 %	38 %	14 %	38 %	12.8	13.3	13.5	14.3	17 %	26 %	23 %	35 %	10.3	10.3	10.0	10.4
PCFG	2.2 %	22 %	3.9 %	29 %	10.2	11.9	10.6	12.0	16 %	21 %	15 %	24 %	9.7	9.8	8.3	8.7
Hybrid	16 %	27 %	14 %	29 %	12.8	13.0	13.5	13.9	17 %	25 %	23 %	34 %	10.3	10.3	10.0	10.4
Combined	13 %	36 %	13 %	37 %	12.8	13.2	13.2	14.3	17 %	27 %	22 %	36 %	10.3	10.3	9.5	10.5

be similar. The result from the table confirms that if the thread model is online guessing attacks in which the number of attempts allowed by an adversary is limited, GNG cannot accurately measure the crack-resistency of PRAs, and β-RSG is a more appropriate metrics in this use case. The low marginal effect in β-RSG also indicates that websites might not need to reject too many passwords if the major concern is online guessing attacks.

From Fig. 2, perhaps the most noticeable difference among the metrics is the relative order of the PCFG method, two versions of *zxcvbn*, and the Hybrid method, comparing with the other Client-end PRAs.

The PCFG method performs reasonably well in GNG, but poorly in β-RSG and β-NRSG. While PCFG can cover many passwords in the testing datasets, which leads to the low total density of passwords not cracked in GNG, some of the high-frequency passwords remain uncovered. As a result, the residual strength of PCFG is lower than most of the other PRAs.

On the other hand, the hybrid method and $zxcvbn_2$ perform much better in β-RSG and β-NRSG than in GNG. Although the high-ranking passwords in the PRAs only include a relative low number of unique passwords in the testing datasets, the popularly selected passwords are mostly covered. Therefore, after rejecting the top-ranking passwords from the PRAs, an adversary can only break into a limited number of accounts within a small number of guesses, which results in a high residual strength.

Another observation is that the performance of the two *zxcvbn* PRAs, especially $zxcvbn_2$ significantly boost in β-NRSG comparing with that in β-RSG. The residual strength resulted by $zxcvbn_2$ is even higher than the size-limited Markov Models (MC_3 and MCB_c). The observation indicates that the relative poor performance of *zxcvbn* in β-RSG is mainly due to the penalization from the large number of passwords, which are extremely not likely to be used, generated.

Overall, several Server-end PRAs including the Blacklist PRA, the Markov Models, and the Combined method result in similar performances. The hybrid method, which inherits the advantage of Blacklist PRAs and Markov Model with backoff, outperform the other Client-end PRAs.

6 Conclusion

In this paper, we model different password strength checking methods (including password strength meters) as Password Ranking Algorithms (PRAs), and we introduce two metrics: the β-Residual Strength Graph (β-RSG) and the Normalized β-Residual Strength Graph (β-NRSG), to compare them using real world password datasets. In our evaluation, we find unreasonably high frequency of some suspicious passwords. We remove the associated accounts by identifying suspicious account IDs. We then, apply the metrics on cleansed datasets, and show that dictionary-based PRA has similar performance with the sophisticated PRAs. If the size of PRAs are limited in order to be fit into a client, a hybrid method combining a small dictionary of weak passwords and a Markov Model with backoff with a limited size can provide the most accurate strength measurement.

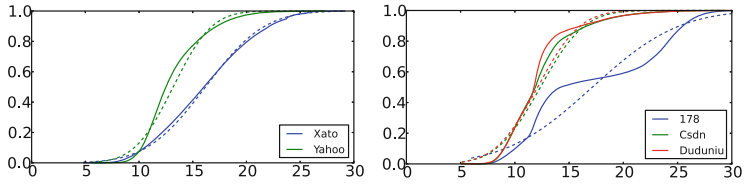
Acknowledgement. This paper is based upon work supported in part by an IBM OCR grant from IBM Research.

A DBSCAN

DBSCAN [17] is a density-based clustering algorithm. It groups together points that are closely packed together. DBSCAN requires two parameters: ϵ and the minimum number of points required to form a dense region $minPts$. It starts with an arbitrary starting point that has not been visited. This point's $\epsilon-$neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as noise. Note that this point might later be found in a sufficiently sized $\epsilon-$environment of a different point and hence be made part of a cluster. If a point is found to be a dense part of a cluster, its $\epsilon-$neighborhood is also part of that cluster. Hence, all points that are found within the $\epsilon-$neighborhood are added, as is their own $\epsilon-$neighborhood when they are also dense. This process continues until the density-connected cluster is completely found. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise. Please refer to [17] for more details.

B Lognormal Distribution in Entropy of IDs

Figure 3 shows CDF of $-\log_{10} p$ of all IDs with length 10, which is the most frequently chosen ID length, from the 5 datasets containing IDs. For each dataset,



(a) English datasets (b) Chinese datasets

Fig. 3. CDF of $-\log_{10} p$. The dashed lines are CDF of normal distribution with the same mean and standard deviation

we calculate probabilities of IDs utilizing 5-order Markov Model trained on itself. The dashed lines in the graph illustrate CDF of normal distribution with the same mean and standard deviation as the corresponding distribution. The figures show that except *178* dataset, the distributions fits relatively well, especially for English datasets. The difference between the distribution from Chinese datasets, *178* in particular, and the corresponding normal distribution is larger, we believe the distribution is biased by the problematic IDs.

C Zxcvbn Password Generation and Modification

Password Generation. *Zxcvbn* was designed to evaluate password strength only. We implemented a password generator following the logic, which takes an input as the maximum entropy, and generate all passwords whose entropy is less than that value. The password generation is a recursive depth-first search process. We start from an empty string. In each iteration, we append a chunk to the current string. If the current entropy is less than the maximum entropy allowed, we record the password, and continue the recursion process. Note that we might duplicated generate passwords as each password might have multiple ways to be decomposed into patterns. Therefore, after the password generation, we conduct a further post-processing step. If a password appears multiple times, we keep the one with the lowest entropy, and then sort all the unique passwords based on entropy.

In practice, an integer score from 0 to 4 is calculated from entropy and each value is assigned with a description (e.g., Weak, Medium, Strong). Passwords with entropy less than 20 are assigned with a score is 0, and are usually rejected. We first tried to create all passwords within 20-bits of entropy. However, after 1 billion attempts, we still have not finished generating passwords start from "mary", which is the first word in female names dictionary. The number of passwords considered as weak by *zxcvbn* is much larger than any of the known weak password dictionary. Alternatively, we generated 10,478,853 unique passwords with entropy less than 4 for $zxcvbn_1$, and 1,834,980 unique passwords with entropy less than 12 for $zxcvbn_2$.

Adapting Zxcvbn to Chinese Datasets. *Zxcvbn* is originally designed for English speaking users, as it supports English words only [38]. In order to adapt

the method for evaluating Chinese passwords, we create another dictionary for evaluating Chinese passwords. We construct such a Chinese dictionary using the *Duduniu* dataset. For each password in *Duduniu*, we first split the word into chunks based on character types, e.g. letters, digits, and symbols. We then, count the frequency of letter chunks after turning all letters into lower cases. Finally, we generate the dictionary by outputing all the letter chunks that contains at least three characters and with frequencies of at least 100 in the descending order of their frequency. There are 5,553 words in the dictionary.

Table 7. Top 20 words in the new dictionary for *zxcvbn*.

Rank	Words									
1–10	asd	woaini	wang	abc	zhang	liu	qwe	love	qaz	yang
11–20	chen	zxc	aaa	wei	www	long	lin	xiao	aaaaaa	huang

Table 7 lists the first 20 words in order. Most of the common words used are the syllables of last names in Chinese, e.g. wang, zhang, liu, etc. The rest of them are either keyboard patterns or letter sequences. Not many English words appears in the dictionary. There are many three letter combinations in the dictionary, such as wjq, ljh, zjh. We believe these are initials of the syllables in Chinese names. There is no need to construct separate name dictionaries as the most common names are already covered.

We were able to generate 9,316,973 passwords with entropy less than 3 for $zxcvbn_1$, and 1,913,061 unique passwords with entropy less than 12 for $zxcvbn_2$.

References

1. Passwords (2009). http://wiki.skullsecurity.org/Passwords
2. CSDN cleartext passwords (2011). http://dazzlepod.com/csdn/
3. John the ripper password cracker (2014). http://www.openwall.com/john/
4. Adams, A., Sasse, M.A.: Users are not the enemy. Commun. ACM **42**(12), 40–46 (1999)
5. Bergadano, F., Crispo, B., Ruffo, G.: Proactive password checking with decision trees. In: Proceedings of the 4th ACM Conference on Computer and Communications Security, pp. 67–77 (1997)
6. Bonneau, J.: The science of guessing: analyzing an anonymized corpus of 70 million passwords. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 538–552 (2012)
7. Bonneau, J., Herley, C., van Oorschot, P.C., Stajano, F.: Passwords and the evolution of imperfect authentication. Commun. ACM **58**(7), 78–87 (2015)
8. Boztas, S.: Entropies, guessing, and cryptography. Technical report 6, Department of Mathematics, Royal Melbourne Institute of Technology (1999)
9. Brostoff, S., Sasse, M.A.: "Ten strikes and you're out": increasing the number of login attempts can improve password usability. In: Proceedings of the Human-computer Interaction Security Workshop (2003)

10. Burnett, M.: Today I am releasing ten million passwords (2015). https://xato.net/passwords/ten-million-passwords/
11. Burr, W.E., Dodson, D.F., Polk, W.T.: Electronic authentication guideline. US Department of Commerce, Technology Administration, National Institute of Standards and Technology (2004)
12. Castelluccia, C., Chaabane, A., Dürmuth, M., Perito, D.: When privacy meets security: Leveraging personal information for password cracking. arXiv preprint arXiv:1304.6584 (2013)
13. Castelluccia, C., Dürmuth, M., Perito, D.: Adaptive password-strength meters from Markov models. In: Proceedings of the Network and Distributed System Security Symposium (2012)
14. de Carné de Carnavalet, X., Mannan, M.: From very weak to very strong: analyzing password-strength meters. In: Proceedings of the Network and Distributed System Security Symposium (2014)
15. Dell'Amico, M., Filippone, M.: Monte carlo strength evaluation: fast and reliable password checking. In: Proceedings of the 22nd ACM Conference on Computer and Communications Security, pp. 158–169 (2015)
16. Egelman, S., Sotirakopoulos, A., Muslukhov, I., Beznosov, K., Herley, C.: Does my password go up to eleven?: the impact of password meters on password selection. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2379–2388 (2013)
17. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd ACM Conference on Knowledge Discovery and Data Mining, vol. 96, pp. 226–231 (1996)
18. Florêncio, D., Herley, C.: A large-scale study of web password habits. In: Proceedings of the 16th International Conference on World Wide Web, pp. 657–666 (2007)
19. Forget, A., Chiasson, S., van Oorschot, P.C., Biddle, R.: Improving text passwords through persuasion. In: Proceedings of the 4th Symposium on Usable Privacy and Security, pp. 1–12 (2008)
20. Grampp, F.T., Morris, R.H.: The unix system: unix operating system security. AT&T Bell Laboratories Tech. J. **63**(8), 1649–1672 (1984)
21. Herley, C., van Oorschot, P.C.: A research agenda acknowledging the persistence of passwords. IEEE Secur. Priv. **10**(1), 28–36 (2012)
22. Kelley, P.G., Komanduri, S., Mazurek, M.L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L.F., Lopez, J.: Guess again (and again and again): measuring password strength by simulating password-cracking algorithms. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 523–537(2012)
23. Klein, D.V.: Foiling the cracker: a survey of, and improvements to, password security. In: Proceedings of the 2nd USENIX Security Workshop, pp. 5–14 (1990)
24. Komanduri, S., Shay, R., Cranor, L.F., Herley, C., Schechter, S.: Telepathwords: preventing weak passwords by reading users' minds. In: Proceedings of the 23rd USENIX Security Symposium, pp. 591–606 (2014)
25. Komanduri, S., Shay, R., Kelley, P.G., Mazurek, M.L., Bauer, L., Christin, N., Cranor, L.F., Egelman, S.: Of passwords and people: measuring the effect of password-composition policies. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2595–2604 (2011)
26. Ma, J., Yang, W., Luo, M., Li, N.: A study of probabilistic password models. In: IEEE Symposium on Security and Privacy (SP), pp. 689–704. IEEE (2014)

27. Manber, U., Wu, S.: An algorithm for approximate membership checking with application to password security. Inf. Process. Lett. **50**(4), 191–197 (1994)
28. Morris, R., Thompson, K.: Password security: a case history. Commun. ACM **22**(11), 594–597 (1979)
29. Narayanan, A., Shmatikov, V.: Fast dictionary attacks on passwords using time-space tradeoff. In Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 364–372 (2005)
30. Riley, S.: Password security: What users know and what they actually do. In: Chaparro, B.S. (ed.) Usability News, vol. 8 of 1, Software Usability Research Laboratory (SURL) at Wichita State University (2006)
31. Schechter, S., Herley, C., Mitzenmacher, M.: Popularity is everything: a new approach to protecting passwords from statistical-guessing attacks. In: Proceedings of the 5th USENIX Conference on Hot Topics in Security, pp. 1–8 (2010)
32. Spafford, E.H.: OPUS: preventing weak password choices. Comput. Secur. **11**(3), 273–278 (1992)
33. Ur, B., Kelley, P.G., Komanduri, S., Lee, J., Maass, M., Mazurek, M., Passaro, T., Shay, R., Vidas, T., Bauer, L., et al.: How does your password measure up? The effect of strength meters on password creation. In: Proceedings of the 21st USENIX Security Symposium, pp. 65–80 (2012)
34. Ur, B., Segreti, S.M., Bauer, L., Christin, N., Cranor, L.F., Komanduri, S., Kurilova, D., Mazurek, M.L., Melicher, W., Shay, R.: Measuring real-world accuracies and biases in modeling password guessability. In: Proceeding of the 24th USENIX Security Symposium, pp. 463–481 (2015)
35. Veras, R., Collins, C., Thorpe, J.: On the semantic patterns of passwords and their security impact. In: Proceedings of the Network and Distributed System Security Symposium (2014)
36. Weir, M., Aggarwal, S., Collins, M., Stern, H.: Testing metrics for password creation policies by attacking large sets of revealed passwords. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 162–175 (2010)
37. Weir, M., Aggarwal, S., de Medeiros, B., Glodek, B.: Password cracking using probabilistic context-free grammars. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 391–405 (2009)
38. Wheeler, D.: zxcvbn: realistic password strength estimation. Dropbox blog article (2012)
39. Yan, J.J.: A note on proactive password checking. In: Proceedings of the 2001 Workshop on New Security Paradigms, pp. 127–135 (2001)