# Understanding Passwords of Chinese Users: Characteristics, Security and Implications

Ding Wang *Student Member, IEEE*, Haibo Cheng, Qianchen Gu, Ping Wang *Senior Member, IEEE*

**Abstract**—While a lot has changed in computer security over the last twenty years, textual passwords remain the dominant mechanism of authentication over computer systems and are likely to persist in the foreseeable future. Though much attention has been paid to passwords chosen by English users, relatively little is known about passwords selected by non-English users, especially by those who use hieroglyphic characters as their native languages. In this work, we initiate a systematic study into the fundamental properties that characterize passwords of Chinese users, the largest Internet population in the world. We, for the first time, uncover several striking findings on the basis of a corpus of 100 million real-life Chinese web passwords (as well as 30 million English ones), the largest corpus of user-generated passwords ever studied.

We further conduct a series of experiments on these datasets by employing two state-of-the-art password cracking techniques, i.e., PCFG-based ones and Markov-Chain-based ones. Remarkably, our results reveal a "reversal principle": when the guess number allowed is small, Chinese web passwords are much weaker than their English counterparts, yet this relationship will be reversed when the guess number is large. This implies that at somewhere these two groups will be of similar security strength, which well reconciles two conflicting claims about the strength of Chinese web passwords made by Bonneau in IEEE S&P'12 and Li et al. in USENIX SEC'14, respectively. At ten million guesses, the success rate of our improved PCFG-based attack (using Duowan as the training set) against the remaining five Chinese datasets is from 33.2% to 49.8%, which means that our improved attack can crack 92% to 188% more passwords than the best record reported by Li et al. in 2014. We believe that our work contributes to a much better understanding of the characteristics and security of Chinese passwords, and will serve as a groundwork for world-wide security administrators and Chinese individual users to more informedly secure their service accounts.

**Index Terms**—User authentication, Password security, Semantic pattern, Probabilistic context-free grammar, Markov model.

## 1 Introduction

*Growth begins only when we begin to accept our own weakness.* –Jean Vanier. *Befriending the Stranger*

Password authentication is the dominant form of access control in computer systems. Though the security pitfalls of textual passwords were revealed as early as four decades ago [1] and various alternative authentication methods (e.g., graphical passwords [2] and multi-factor authentication [3]) have been proposed since then, they stubbornly survive and reproduce with every new computer system while Internet technologies have advanced by leaps and bounds in other areas. For one reason, textual passwords offer advantages, such as low deployment cost, easy recovery and remarkable simplicity, which cannot always be matched by their alternatives [4]. The matter is further complicated by the shortage of effective methods to quantify the obscure costs of replacing passwords [5], while marginal gains are often insufficient to reach the activation energy that is necessary to overcome the significant transition costs. Thus, passwords remain the primary method for user identification and are likely to persist in the foreseeable future.

Despite its ubiquity, password authentication is stuck with an inherent tension [6]: truly random passwords are hard for users to memorize, while user-memorable passwords may be highly predictable. To deal with this notorious "security-usability" dilemma, researchers have devoted significant efforts [7]–[10] to the following two types of studies. *Type-1* research aims at evaluating the strength of a password dataset by gauging its Shannon entropy as in NIST-800-63 [11], or by measuring its "guessability". The latter notion characterizes the fraction of passwords that, at a given number of guesses, can be cracked by password cracking algorithms like the probabilistic context-free grammars (PCFG) [12] and Markov-Chain-based attacks [8].

*Type-2* research attempts to reduce the use of weak passwords, and mainly two approaches have been utilized: proactive password checking [13] and password strength meter [14]. The former checks the user-selected passwords and only accepts the ones that comply with the system policy (e.g., at least 8 characters long). The latter is typically a visual feedback of password strength, often presented as a colored bar to help users create stronger passwords [15]. Most of today's leading sites employ a combination of these two approaches to restrict users from choosing weak passwords. In this work, though we mainly focus on type-1 research, one can see that our results are also helpful for type-2 research.

### 1.1 Motivations

Existing literature mainly focuses on passwords chosen by English users, or more precisely, by netizens that use the Latin alphabet, yet little attention has been paid to the characteristics and strength of passwords generated by those who use other native languages. For instance, password "`wanglei123`" is currently deemed "Strong" by many high-profile password strength meters such as those of AOL, Google, IEEE, and Sina weibo (known as "Chinese Twitter"). However, there is no doubt that this password is highly prone to guessing, for "`wanglei`" is an extremely common Chinese Pinyin (i.e., the Chinese phonetic alphabet) name but not a random string of length seven. Failing to catch this is equal to blindly overlooking the weaknesses of Chinese passwords, posing the corresponding account at high risks.

• *D. Wang, H. Chen, Q. Gu and P. Wang are with the School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China. Email: {wangdingg, hbchen, cqgu, pwang}@pku.edu.cn*

*Manuscript received 27 April 2015; revised .*

Not surprisingly, so far there has been no satisfactory answer to the following fundamental questions: *Are there any characteristics that differentiate Chinese web passwords from English web passwords? What's the security strength of Chinese web passwords? Are they weaker or stronger than English web passwords?* As there have been 649 million Chinese Internet users by the end of 2014 [16], which account for more than a quarter (and also the largest proportion) of the world's Internet population, and this number still grows with its momentum, it is of great importance to answer the aforementioned questions to provide both security administrators and Chinese users with necessary security guidance. For instance, if the answer to the first question is affirmative, then it highly indicates that the traditional password creation policies and strength meters originally designed for English users can not be readily applied to Chinese users.

To the best of our knowledge, Li et al.'s work [17] may be the closest to what we will investigate in the current paper. However, as we will show in this work, it imprudently renders two of its five Chinese datasets at best useless and at worst negative when performing data cleansing. As a result, the effectiveness of the results reported in [17] will be greatly impaired. Besides this, many fundamental characteristics, such as length distribution, frequency distribution and Pinyin-name-based semantic patterns, have not been explored in their work. What's more, they proposed an improved PCFG-based cracking algorithm and at 10 billion guesses, their best success rate is about 17.3%, which is significantly lower than what will be reported based on our improved PCFG-based algorithm in this work. Compared with our attacking results based on the Markov-based algorithm, this gap will be even more prominent. *Most importantly*, Li et al. invalidated the remarks made in [7] that Chinese passwords are among the strongest ones to guess than those of many other populations like English users, and further concluded that "the strength of the passwords of Chinese and English users is similar". However, we will demonstrate that Li et al.'s conclusion is still biased. Last but not the least, no implications of the characteristics and strength of Chinese web passwords has been explored in [17], and we will fill the gap in this work.

We also note that, Ma et al.'s work [8] has employed six password datasets, three of which are from Chinese websites, yet it mainly deals with the effectiveness of probabilistic password cracking models and pays little attention to the characteristics of Chinese web passwords.

### 1.2 Contributions

To answer the above three fundamental questions, we leverage about 100 million publicly available passwords from six popular Chinese sites and more than 30 million passwords from three English sites, the largest corpus of user-chosen passwords ever studied. Benefiting from the plain-text form of these 130 million passwords, we seek for fundamental properties that characterize user-chosen passwords, and for the first time manage to identify several striking characteristics of Chinese web passwords in comparison to English web passwords. Particularly, by investigating the password character distributions in terms of inversion number, we are able to provide a better understanding of to what extent user passwords are influenced by their native languages.

As Chinese users are familiar with Pinyin and each has a Pinyin name (e.g., "`wanglei`" and "`zhangwei`"), is there a high probability that they insert their Pinyin names into their passwords? We establish it in this work: *every one in nine*. Surprisingly, the proportion of English passwords that include an English name (at least five characters long) reaches a staggering 24.3%, i.e., one in four. To the best of our knowledge, we for the first time explore the name patterns in a large-scale empirical password study. Further, we confirm that English users excessively employ common English words to build passwords. In contrast, being incompetent to understand English words, Chinese users excessively employ digits, especially dates and simple digit patterns — every one in six users inserts a 6-digit birthdate into her password. Surprisingly, despite these notable differences in password composition, passwords chosen by these two distinct groups of users (i.e., in terms of language and location) are of quite similar frequency and length distributions.

While there are distinctive characteristics between passwords of English users and Chinese users, an interesting question naturally arises: which group of passwords is generally more secure? In this work, we employ two state-of-the-art password-cracking algorithms (i.e., PCFG-based and Markov-Chain-based [8]) to measure the strength of Chinese web passwords. Based on the identified characteristics of Chinese passwords, we improve the original PCFG-based algorithm to more accurately capture user-generated passwords that are of a monotonically long structure (e.g., "`1qa2ws3ed`" and "`1a2b3c4d`"). We further adjust the PCFGs learned from the training set by adding about 98.6K Chinese Pinyin names and 24.4K six-digit birthdays. At ten million guesses, our improved approach is able to crack 92% to 188% more passwords than the results reported by Li et al. [17] in 2014. Remarkably, our extensive experiments and comprehensive comparisons reveal a "*reversal principle*": when the guess number allowed is small, Chinese passwords are much weaker than their English counterparts, yet this relationship will be reversed when the guess number is large, thereby reconciling the contradictory claims made in [7], [17].

Furthermore, we highlight some critical implications that our above findings are likely to have for password cracking, password strength meters and password creation policies. As far as we know, we, for the first time, provide a large-scale empirical evidence (i.e., on the basis of 6.43 million CSDN passwords and 16.26 million Dodonew passwords) supporting for the hypothesis proposed in [15], [18] that users rationally choose more complex and secure passwords for accounts associated with higher value.

## 2 Related work

It has long been an interesting (yet challenging) research area to analyze user-generated passwords, dating back at least to Morris and Thompson's seminal work in 1979 [1]. They analyzed a corpus of 3,000 passwords and reported some basic password statistics like password lengths (e.g., 71.12% were no more than 6 characters long) and frequency of non-alphanumeric characters (e.g., 13.93% of passwords). This work has been followed by a great number of studies, some notable ones include [7], [12], [14], [19], [20]. We present an overview of prior research on password characteristics and cracking, while other topics such as password usability and management [21], [22] are out of the scope of this work.

### 2.1 Password characteristics

The literature features a wealth of analysis of password characteristics on English web passwords, including statistics on structural patterns and much deeper semantic patterns.

**Basic statistics**. In 1990, Klein [19] collected 13,797 computer accounts from his friends and acquaintances around US and UK, and observed that users tend to choose passwords that can be easily derived from dictionary words: a dictionary of 62,727 words is able to crack 24.21% of the collected accounts and 51.70% of the cracked passwords are shorter than 6 characters long. In 2004, Yan et al. [6] found that user-chosen passwords are likely to be dictionary words since users have difficulty in memorizing random strings, and that the lengths of passwords in their study (involving 288 participants) are on average between 7 and 8.

In 2012, Bonneau [7] conducted a systematic analysis of 70 million Yahoo passwords. This work examines dozens of subpopulations based on demographic factors (e.g., age, gender and language) and site usage characteristics (e.g., email and retail), and finds that "even seemingly distant language communities choose the same weak passwords". Particularly, Chinese passwords are among the most difficult ones to crack. In 2014, however, Li et al. [17] argued that Bonneau's dataset is not representative of general Chinese users, for Yahoo users are those who are familiar with English. Accordingly, Li et al. leveraged a corpus of five datasets from Chinese sites and observed that Chinese users like to use digits when creating passwords, as compared to English users who like to use letters to build passwords. However, as an elementary defect, two of their Chinese datasets have not been cleaned properly (see Section 3.2), which might lead to inaccurate measures and biased comparisons. More importantly, several critical password properties (such as the most popular passwords, the length and frequency distributions of passwords) remain to be explored.

In 2014, Ma et al. [8] investigated password characteristics about the length and the structure of six datasets, three of which are from Chinese websites. Nonetheless, this work mainly focuses on the effectiveness of probabilistic password cracking models and pays little attention to the deeper semantic patterns of passwords (e.g., no information is provided about the role of Pinyins, names or dates).

**Semantic patterns**. In 1989, Riddle et al. [23] found that birth dates, personal names, nicknames and celebrity names are common in user-chosen passwords. In 2004, Brown et al. [24] confirmed this by conducting a thorough survey that involved 218 participants and 1,783 passwords, and they reported that the most frequent entity in passwords is the self (accounts for 66.5%), followed by relatives (7.0%), lovers and friends; also, names (32.0%) were found to be the most common information used, followed by dates (7.2%). In 2012, Veras et al. [25] examined the 32 million RockYou dataset by employing visualization techniques and observed that 15.26% of passwords contain sequences of 5-8 consecutive digits, 38% of which could be further classified as dates. They also found that repeated days/months and holidays are popular, and when non-digits are paired with dates, they are most commonly single-characters, or names of months.

In 2014, Li et al. [17] noted that Chinese users tend to insert Pinyins and dates into their passwords. However, many other important semantic patterns (e.g., Pinyin name, place and mobile number) are left unexplored. In addtion, as we will show in this work, due to an imprudent processing of data cleansing and inappropriate choices of password cracking algorithms, Li et al.'s measurement of the strength of Chinese passwords is largely biased.

### 2.2 Password cracking

A crucial research area in password cryptography is to study the security strength of user-chosen passwords. To avoid using the brute-force attack, earlier work (e.g., [19], [23]) uses a combination of ad hoc dictionaries and mangling rules to model the common password generation practice and see whether user passwords can be successfully rebuilt in a period of time. This technique greatly improves the cracking efficiency and has given rise to automated password cracking tools such as John the Ripper (JTR) [26].

Borrowing the idea of Shannon entropy, the NIST Electronic Authentication Guideline [11] attempts to use the concept of *password entropy* for estimating the strength of password creation policy underlying a password system. Password entropy is calculated mainly according to the length of passwords. Florencio and Herley [27], and Egelman et al. [15] improved this approach by adding the size of the alphabet into the calculation and called the resulting value $log_2((alpha.size)^{pass.len})$ the bit length of a password.

However, previous ad hoc metrics (e.g., password entropy and bit length) have recently been shown far from accurate by Weir et al. [28], and they suggested that the approach based on simulating password cracking sessions is more promising. They also developed a novel method that first automatically derives word-mangling rules from password datasets, and then instantiates the derived grammars by using string segments from external input dictionaries (e.g., the famous Dic-0294 wordlists) to generate guesses in decreasing probability order [12]. This PCFG-based cracking approach is able to crack 28% to 129% more passwords than JTR [26] when given the same number of guesses. It has been considered as one of the state-of-the-art password cracking techniques and used in a number of recent works [8], [10].

Different from the PCFG-based approach, Narayanan and Shmatikov [29] suggested a template-based model in which the Markov-Chain theory is used for assigning probabilities to letter-based string segments, and it substantially reduces the password search space. This approach was tested in an experiment against 142 real-life user passwords and was able to break 67.6% of them. In 2014, by using various normalization and smoothing techniques from the natural language processing (NLP) domain, Ma et al. [8] systematically evaluated the Markov-based model and found that, in some cases, it performs significantly better than the PCFG-based model at large guesses (e.g., $2^{30}$) when parameterized appropriately. In this work, we will perform extensive experiments by using both attacking models (combined with the identified characteristics of Chinese passwords) to evaluate the strength of Chinese passwords.

## 3 Characteristics of Chinese passwords

In this section, we first describe the six Chinese web password datasets and three English password datasets, a total corpus of 130 million passwords that will be used in our analysis and later experiments, and then reveal several important findings about Chinese password characteristics.

TABLE 1
Data Cleansing of the nine datasets

| Dataset | Web service | Location | Language | Original | Miscellany | Length>30 | All removed | After cleansing | Unique PWs |
|---|---|---|---|---|---|---|---|---|---|
| Tianya | Social forum | China | Chinese | 31,761,424 | 860,178 | 5 | 2.71% | 30,901,241 | 12,898,437 |
| 7k7k | Gaming | China | Chinese | 19,138,452 | 13,705,087 | 10,078 | 71.66%* | 5,423,287 | 2,865,573 |
| Dodonew | Gaming&E-commerce | China | Chinese | 16,283,140 | 10,774 | 13,475 | 0.15% | 16,258,891 | 10,135,260 |
| 178 | Gaming | China | Chinese | 9,072,966 | 0 | 1 | 0.00% | 9,072,965 | 3,462,283 |
| CSDN | Programmer forum | China | Chinese | 6,428,632 | 355 | 0 | 0.01% | 6,428,277 | 4,037,605 |
| Duowan | Gaming | China | Chinese | 5,024,764 | 42,024 | 10 | 0.83% | 4,982,730 | 3,119,060 |
| Rockyou | Social forum | USA | English | 32,603,387 | 18,377 | 3140 | 0.07% | 32,581,870 | 14,326,970 |
| Yahoo | Portal(e.g., E-commerce) | USA | English | 453,491 | 10,657 | 0 | 2.35% | 442,834 | 342,510 |
| Phpbb | Programmer forum | USA | English | 255,421 | 45 | 3 | 0.02% | 255,373 | 184,341 |

*We remove 13M duplicate accounts from 7k7k, because we identify that they are copied from Tianya as we will detail in Section 3.2.

### 3.1 Ethics consideration and dataset descriptions

The nine datasets described here are different in terms of service, size, language and user localization (see Table 1). They were hacked by external attackers or disclosed by anonymous insiders, and were subsequently made public on the Internet. We realize that though publicly available, these datasets are private data. Therefore, we only report the aggregated statistical information, and treat each individual account as confidential such that using it in our research will not increase risk to the corresponding victim, i.e., no personally identifiable information can be learned. Furthermore, these datasets may be exploited by attackers as training sets or cracking dictionaries, while our use of them is both beneficial for the academic community to understand password choices of Chinese netizens and for security administrators to secure user accounts.

The first three datasets are all from US. The Rockyou dataset [30] includes over 32M passwords and was hacked from the social application site rockyou.com in Dec. 2009 by an SQL injection attack. The Phpbb dataset contains around 255K passwords leaked from phpbb.com, a forum on the development of PHP scripting language, in Jan. 2009. The Yahoo dataset [31] consists of about 442K passwords leaked by the hacker group named D33Ds in July 2012.

The following six datasets were all leaked from Chinese sites in Dec. 2011 when a series of security breaches happened [32], and we collected them at that time. The 6.42M CSDN passwords were hacked from csdn.net, a popular community for software developers in China. The 31.76M Tianya passwords were leaked from tianya.cn, an influential Chinese BBS forum. The remaining four datasets are all from popular Chinese gaming websites, of which the Dodonew dataset deserves attention, for it is from a site with e-commerce services and its accounts are generally perceived to be of important value. As expected, this dataset is the strongest one among all datasets (see Section 4). Note that three pairs of datasets (i.e., Tianya vs. Rockyou, Dodonew vs. Yahoo, and CSDN vs. Phpbb) will be used for strength comparison in Section 4, for each pair is of the same service.

### 3.2 Data cleansing

Before examining the password characteristics, we perform the process of data cleansing. We get rid of email addresses and user names from the original data. We note that the remaining data consists of some strings whose lengths are over 100 (e.g., there are dozens of passwords in Rockyou with a length up to 128), which highly indicates that they are junk information. We remove such strings and others that include characters beyond the 95 printable ASCII symbols. We further remove strings whose length is over 30, because after having manually scrutinized the original datasets, we find that these long strings do not seem to be generated by human beings, but more likely by password managers. Moreover, such unusually long passwords are often beyond the scope of attackers who specially care about cost-effectiveness. Though the fraction of excluded passwords is negligible, this step of cleansing unifies the input and largely simplifies the later data processing.

Of particular importance is our observation that, the Tianya dataset and 7k7k dataset largely overlap with each other. We were first puzzled by the fact that the password "`111222tianya`" originally lay in the top-10 most popular list of both datasets. We manually scrutinize the original datasets (i.e., before removing the email addresses and user names) and are surprised to find that there are around 3.91 million (actually 3.91*2 million due to a split representation of 7k7k accounts, as we will discuss later) joint accounts in both datasets, and we realize that someone probably have copied these joint accounts from one dataset to the other.

Now, a natural question arises: *From which dataset have these joint accounts been copied?* We believe that these joint accounts were copied from Tianya to 7k7k mainly for two reasons. Firstly, it is unreasonable for 0.34% users in 7k7k to insert the string "`tianya`" into their 7k7k passwords. The following second reason is quite subtle yet convincing. In the original Tianya dataset, we find that the joint accounts are of the form {user name, email address, password}, while in the original 7k7k dataset such a joint account is divided into two parts: {user name, password} and {email address, password}. The password "`111222tianya`" occurs 64822 times in 7k7k and 48871 times in Tianya, and one gets that $64822/2 < 48871$. Therefore, it is more plausible for someone to copy *some* (i.e., $64822/2$ of a total of 48871) accounts using "`111222tianya`" as the password from Tianya to 7k7k, rather than to copy all the accounts (i.e., $64822/2$) using "`111222tianya`" as the password from 7k7k to Tianya and further reproduces $16460 (= 48871 - 64822/2)$ such accounts.

After removing 7.82 million joint accounts from 7k7k, we found that all of the passwords in the remaining 7k7k dataset occur even times (at least two). This is expected, for we observe that in 7k7k half of the accounts are of the form {user name, password}, while the rest are of the form {email address, password}, and it is likely that both forms are directly derived from the form {user name, email address, password}. For instance, both {wanglei, wanglei123} and {wanglei@gmail.com, wanglei123} are actually derived

from the single account {wanglei, wanglei@gmail.com, wanglei123}. Consequently, we further divide 7k7k into two equal parts and discard one part. The detailed information on data cleansing is summarized in Table 1.

As far as we know, Li et al.'s work [17] is the only one that has exploited the datasets Tianya and 7k7k. However, contrary to what we have done in this work, Li et al. think that the 3.91M joint accounts are copied from 7k7k to Tianya. Their only reason is that, when dividing these two datasets into the reused passwords group (i.e., the joint accounts) and the not-reused passwords group, they find that "the proportions of various compositions are similar between the reused passwords and the 7k7k's not-reused passwords, but different with Tianya's not-reused passwords". However, they have never explained what these "various compositions" are. Such vague statements also cannot answer the critical question: why are there so many 7k7k users using "`111222tianya`" as their passwords? Hence, we believe that they should have removed 3.91*2 million joint accounts from 7k7k but not 3.91 million ones from Tianya. In addition, they fail to note that all the passwords in 7k7k occur even times, which is extremely abnormal. As a result, Li et al. render two of their five Chinese datasets at best useless and at worst negative, because contaminated data would highly lead to inaccurate results and unreliable comparisons.

### 3.3 Password characteristics

It is widely hypothesized that user-generated passwords are greatly influenced by their native languages, yet so far little empirical evidence has been given. To fill this gap, here we first illustrate the character distributions of the nine password datasets, and then measure the closeness of passwords with their native languages in terms of inversion number of the character distributions (in descending order).

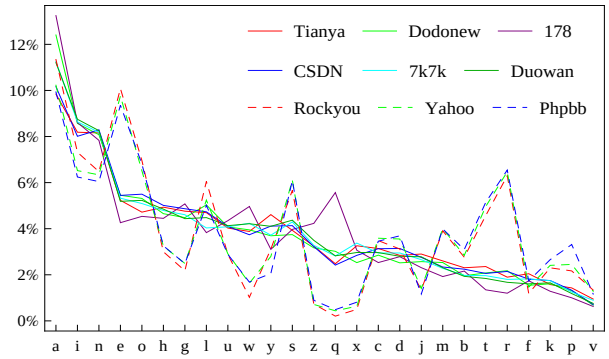



Fig. 1. Letter distributions of user-chosen passwords

Fig. 1 shows that passwords from different language groups have significantly varied letter distributions. What's quite surprising is that, even though generated and used in vastly diversified web services, passwords among the same language group have quite similar letter distributions. This suggests that, when given a password dataset, one can largely determine what's the native language of its users by investigating its letter distribution. Arranged in descending order, the letter distribution of all Chinese passwords is `aineohglwuyszxqcdjmbtfrkpv`, while this distribution for all English passwords is `aeionrlstmcdyhubkgpjvfwzxq`. While some letters (e.g., '`a`', '`e`' and '`i`') occur frequently in both groups, some letters (e.g., '`q`' and '`r`') only occur frequently in one group. Such information can be exploited by attackers to reduce the search space and optimize their cracking strategies. Note that, all the percentages here are handled case-insensitively (e.g., the percentage of letter '`a`' in Tianya is computed as $\frac{\text{\#of occurences of lower/uppercase letter 'a' in Tianya}}{\text{\#of total occurences of all letters in Tianya}} = 11.29\%$).

While users' passwords are greatly affected by their native languages, the letter frequencies in general language usages may not well represent the frequencies of letters that are used in passwords. According to Huang et al.'s work [33], the letter distribution of Chinese language (i.e., written Chinese texts like literary work, newspapers and academic papers), which are converted into Chinese Pinyin, is `inauhegoyszdjmxwqbctlpfrkv`. This shows that some letters (e.g., '`i`' and '`u`'), which is very popular in Chinese user passwords, appear much less frequently in written Chinese texts. Similar observation also holds for English passwords, where the letter distribution of English language (i.e., `etaoinshrdlcumwfgypbvkjxqz`) is obtained from www.cryptograms.org/letter-frequencies.php.

To further explore the closeness of passwords with their native languages and with the passwords from other datasets, we measure the inversion number of the letter distribution sequence (in descending order) between two password datasets (as well as languages), and the results are summarized in Table 2. "Pinyin_fullname" is a dictionary consisting of 2,426,841 unique Chinese full names (e.g., `wanglei` and `zhangwei`), "Pinyin_word" is a dictionary consisting of 127,878 unique Chinese words (e.g., `chang` and `cheng`), and these two dictionaries will be detailed later. Note that the inversion number of sequence $A$ to sequence $B$ is equal to that of $B$ to $A$. For instance, the inversion number of `inauh` to `aniuh` is 3, which is equal to that of `aniuh` to `inauh`.

Table 2 shows that, the inversion number of letter distributions between passwords from the same language group is generally *much smaller* than that of passwords from different language groups. This value is also *distinctly smaller* than that of the letter distributions between passwords and their native language (see the bold values in Table 2). All these indicate that passwords from different languages are intrinsically different from each other in letter distributions, and that passwords are close to their native language yet the distinction is still noticeable (measurable).

Fig. 2 depicts the length distributions of passwords. Irrespective of the web service, language and culture differences, the most common password lengths of every dataset are between 6 and 10, among which length-6 or 8 takes the lead. Merely passwords of these five lengths can account for more than 75% of every entire dataset, and this value will rise to 90% if we consider passwords with lengths of 5 to 12. As expected, very few users prefer passwords longer than 15 characters. Notably, people seem to prefer even length over odd length. Another interesting observation is that, CSDN exhibits only one peak in its length distribution curve and has much fewer passwords (i.e., only 2.16%) in lengths below 8. This is likely due to the fact that this site has enforced a minimal length-8 policy since an early stage.

Fig. 3 portrays the frequency vs. the rank of passwords from different datasets in a log-log scale. We first sort each dataset according to the password frequency in descending

TABLE 2
Inversion number of the letter distributions (in descending order) between the datasets ("PWs" stands for passwords)

| | Tianya | Dodonew | 178 | CSDN | 7k7k | Duowan | All Chinese PWs | Chinese language | Pinyin fullname | Pinyin word | Rockyou | Yahoo | Phpbb | All English PWs | English language |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tianya | 0 | 15 | 22 | 42 | 15 | 17 | 14 | **40** | 32 | 37 | 100 | 100 | 113 | 100 | 99 |
| 7k7k | 15 | 0 | 23 | 31 | 14 | 10 | 13 | **41** | 39 | 38 | 105 | 101 | 112 | 105 | 96 |
| Dodonew | 22 | 23 | 0 | 42 | 21 | 15 | 12 | **52** | 40 | 49 | 94 | 92 | 105 | 94 | 99 |
| 178 | 42 | 31 | 42 | 0 | 41 | 35 | 32 | **56** | 48 | 47 | 134 | 130 | 141 | 134 | 125 |
| CSDN | 15 | 14 | 21 | 41 | 0 | 12 | 15 | **45** | 39 | 42 | 95 | 95 | 106 | 95 | 96 |
| Duowan | 17 | 10 | 15 | 35 | 12 | 0 | 9 | **49** | 39 | 44 | 99 | 97 | 110 | 99 | 98 |
| All_Chinese_PWs | 14 | 13 | 12 | 32 | 15 | 9 | 0 | **44** | 34 | 43 | 104 | 102 | 115 | 104 | 101 |
| Chinese_language | 40 | 41 | 52 | 56 | 45 | 49 | 44 | 0 | 38 | 27 | 118 | 114 | 123 | 118 | 113 |
| Pinyin_fullname | 32 | 39 | 40 | 48 | 39 | 39 | 34 | 38 | 0 | 31 | 124 | 122 | 135 | 124 | 123 |
| Pinyin_word | 37 | 38 | 49 | 47 | 42 | 44 | 43 | 27 | 31 | 0 | 115 | 113 | 124 | 115 | 112 |
| Rockyou | 100 | 105 | 94 | 134 | 95 | 99 | 104 | 118 | 124 | 115 | 0 | 12 | 23 | 0 | **47** |
| Yahoo | 100 | 101 | 92 | 130 | 95 | 97 | 102 | 114 | 122 | 113 | 12 | 0 | 15 | 12 | **39** |
| Phpbb | 113 | 112 | 105 | 141 | 106 | 110 | 115 | 123 | 135 | 124 | 23 | 15 | 0 | 23 | **44** |
| All_English_PWs | 100 | 105 | 94 | 134 | 95 | 99 | 104 | 118 | 124 | 115 | 0 | 12 | 23 | 0 | **47** |
| English_language | 99 | 96 | 99 | 125 | 96 | 98 | 101 | 113 | 123 | 112 | 47 | 39 | 44 | 47 | 0 |

TABLE 3
Top 10 most popular passwords of each dataset

| Rank | Tianya | 7k7k | Dodonew | 178 | CSDN | Duowan | Rockyou | Yahoo | Phpbb |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 123456 | 123456 | 123456 | 123456 | 123456789 | 123456 | 123456 | 123456 | 123456 |
| 2 | 111111 | 0 | a123456 | 111111 | 12345678 | 111111 | 12345 | password | password |
| 3 | 000000 | 111111 | 123456789 | zz12369 | 11111111 | 123456789 | 123456789 | welcome | phpbb |
| 4 | 123456789 | 123456789 | 111111 | qiulaobai | dearbook | 123123 | password | ninja | qwerty |
| 5 | 123123 | 123123 | **5201314** | 123456aa | 00000000 | 000000 | **iloveyou** | abc123 | 12345 |
| 6 | 123321 | **5201314** | 123123 | wmsxie123 | 123123123 | **5201314** | princess | 123456789 | 12345678 |
| 7 | **5201314** | 123 | a321654 | 123123 | 1234567890 | 123321 | 123321 | 12345678 | letmein |
| 8 | 12345678 | 12345678 | 12345 | 000000 | 88888888 | a123456 | rockyou | sunshine | 111111 |
| 9 | 666666 | 12345678 | 000000 | qq66666 | 111111111 | suibian | 12345678 | princess | 1234 |
| 10 | 111222tianya | wangyut2 | 123456a | w2w2w2 | 147258369 | 12345678 | abc123 | qwerty | 123456789 |
| Sum of top10 | 2,297,505 | 440,300 | 533,285 | 793,132 | 670,881 | 338,012 | 669,126 | 4,476 | 7,135 |
| Total accounts | 30,901,241 | 5,423,287 | 16,258,891 | 9,072,965 | 6,428,277 | 4,982,730 | 32,581,870 | 442,834 | 255,373 |
| Percent of top10 | 7.43% | 8.12% | 3.28% | 8.74% | 10.44% | **6.78%** | 2.05% | 1.01% | **2.79%** |

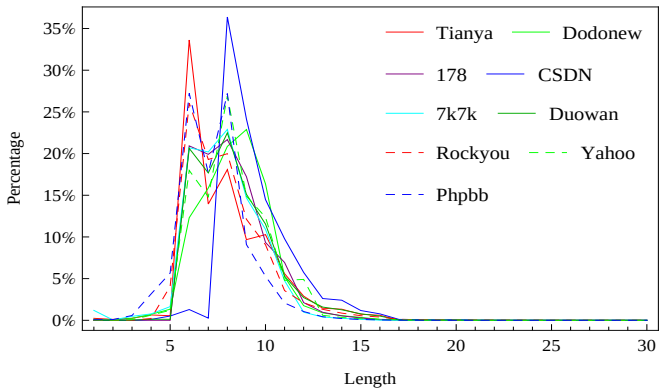



Fig. 2. Length distributions of passwords investigated

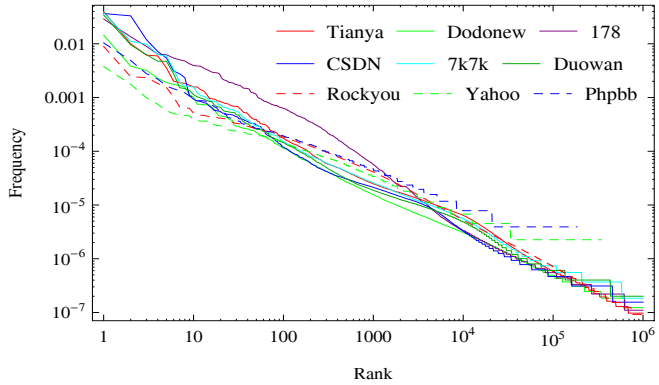



Fig. 3. Frequency distributions of passwords investigated

order, and then each individual password will be associated with a frequency $f_r$ and a rank $r$. Interestingly, the curve for each dataset closely approximates a straight line, and this trend will be more pronounced if we take all the nine curves as a whole. This well corroborates the Zipf theory [34]: $f_r$ and $r$ follow a relationship of the type $\log f_r = \log C - s \cdot \log r$, where $C$ and $s$ are constants. Particularly, $s$ is the absolute value of slope of the Zipf linear regression line and slightly less than 1.0. The Zipf theory indicates that the popularity of user-generated passwords decreases polynomially with the increase of their rank. This further implies that a few passwords are overly popular, while the majority are sparsely scattered in the password space.

Table 3 illustrates the top-10 most frequent passwords from different services. The most frequent password of all datasets is "`123456`", with CSDN being the only exception, which is likely due to the CSDN password policy requiring that no password shall be shorter than 8 characters. "`111111`" follows on the heel. Other popular Chinese passwords include "`123123`", "`123321`", which are all composed of digits and in simple patterns such as repetition and palindrome, while popular ones in English datasets tend to be meaningful letter strings (e.g., the eternal theme of love — frankly, "`iloveyou`" or euphemistically, "`princess`"). Our results confirm the folklore that "back at the dawn of the Web, the most popular password was `12345`. Today, it is one digit longer but hardly safer: `123456`."

It is worth noting that, for each digit, there are many Chinese characters that have similar sounds and this makes it easy to obtain digit sequences that sound like meaningful phrases. Hence, it is unsurprising to see that "`5201314`", which sounds like "I love you forever and ever", ranks the

TABLE 4
Password patterns with digits (The percentage is taken by dividing the corresponding total accounts.)

| Patterns | Tianya | 7k7k | Dodonew | 178 | CSDN | Duowan | Avg. Chinese | Rockyou | Yahoo | Phpbb | Avg. English |
|---|---|---|---|---|---|---|---|---|---|---|---|
| D | **63.77%** | **59.62%** | **30.76%** | **48.07%** | **45.01%** | **52.84%** | **52.93%** | 15.94% | 5.89% | 12.06% | 15.77% |
| LD | 14.71% | 17.98% | 43.50% | 31.12% | 26.14% | 23.97% | 23.72% | **27.70%** | **38.27%** | **19.14%** | **27.78%** |
| Sum of top2 | **78.48%** | **77.60%** | **74.25%** | **79.20%** | **71.15%** | **76.81%** | **76.66%** | 43.64% | 44.16% | 31.20% | 43.55% |

7*th* and 8*th* most popular one in Tianya and Duowan, respectively. Moreover, Chinese passwords are highly concentrated, for only the top-10 most popular ones amount to as high as 6.78%∼10.44% of each entire dataset, with Dodonew being the mere exception. However, even Dodonew achieves 3.24%, while the English datasets are all below 2.80%.

As we have seen that digits are popular in top-10 passwords of Chinese datasets, whether are they popular in the whole datasets? To answer this question, we investigate the frequencies of password patterns that involve *digits*, and results on the top 2 most frequent ones are shown in Table 4. More results (i.e., on the top-10 most frequent ones) can be found in the supplemental material. The first column of the table denotes the pattern of a password as in [12] (i.e., L denotes a lower-case sequence, D for digit sequence, U for upper-case sequence, S for symbol sequence, and the structure pattern of password "`Wanglei123`" is ULD). We see that an average of more than 50% of Chinese web passwords are only composed of digits, while this value of English datasets is only 15.77%. In contrast, English users prefer the pattern LD. Note that, all the percentages hereafter in this work are taken by dividing the corresponding total accounts (e.g., the percentage at the upper-left corner of Table 4 is computed as $\frac{\text{\#of passwords with pattern D}}{\text{\#of total passwords in Tianya}} = \frac{19706174}{30901241} = 63.77\%$).

It is somewhat surprising to see that, the sum of merely the two digit-related patterns (i.e., D and LD) accounts for over 70% of *every* Chinese datasets, indicating that Chinese users excessively employ digits to build their passwords. This is probably due in large part to the fact that most Chinese users are unfamiliar with English language (and Roman letters on the keyboard). If this is the case, is there any meaningful information underlying these digit sequences?

To gain an insight into the underlying semantic patterns, we construct several dictionaries of different semantic categories and investigate their prevalence (see Table 5). "English_word_lower" is from http://www.mieliestronk.com/wordlist.html and it contains about 58,000 popular lower-case English words. "English_lastname" is a dictionary consisting of 18,839 last (family) names with over 0.001% frequency in the US population during the 1990 census, according to US Census Bureau [35]. "English_firstname" contains 5,494 most common first names (1,219 male and 4,275 female names) in US [35]. "English_fullname" is a cartesian product of "English_firstname" and "English_lastname", consisting of about 1.04 million most common English full names.

To get a Chinese full name dictionary, we make use of the 20 million hotel reservations dataset [36] leaked in Dec. 2013. The Chinese family name dictionary includes 504 family names which are officially recognized in China. Since the first names of Chinese users are widely distributed and can be almost any combinations of Chinese words, we do not consider them in this work. As the names are originally in Chinese, we transferred them into Pinyin without tones by using a Python procedure from https://pypinyin.readthedocs.org/en/latest/ and removed the duplicates. We call these two dictionaries "Pinyin_fullname" and "Pinyin_familyname", respectively.

"Pinyin_word_lower" is a Chinese word dictionary known as "SogouLabDic.dic", and "Pinyin_place" is a Chinese place dictionary. Both of them are from [37] and also originally in Chinese, and we translate them into Pinyin in the same way as we tackle the name dictionaries. "Mobile_number" consists of all potential Chinese mobile numbers, which are 11-digit strings with the first seven digits conforming to pre-defined specific values and the last four digits being random. As for the birthday dictionaries, we use patterns to match digit strings that might be birthdays. For example, "YYYYMMDD" stands for a birthday pattern that the first four digits indicate years (from 1900 to 2014), the middle two represent months (from 01 to 12) and the last two denote dates (from 01 to 31). "PW with a $l^+$-letter substring" is a subset of the corresponding dataset (see Table 5) and consists of all passwords that include a letter substring *no shorter than* $l$, and similarly for "PW with a $l^+$-digit substring".

Table 5 shows the various semantic patterns existing in Chinese and English web passwords. We can see that, a large fraction of English users tend to use raw English words as their password building blocks. More specially, 25.88% English users insert a 5-letter or longer (denoted by $5^+$-letter) *word* into their passwords, and this figure accounts for more than a third of the total passwords with a $5^+$-letter substring. In contrast, few Chinese users choose raw Pinyin words or English words to build passwords, yet they prefer Pinyin names, especially *full names*. Surprisingly, of all the Chinese passwords (22.42%) that include a $5^+$-letter substring, more than half (11.24%) include a $5^+$-letter Pinyin full name. There is even a non-negligible proportion (i.e., 4.10%) of English passwords that contain a $5^+$-letter full Pinyin name, and a reasonable explanation is that many Chinese users have created accounts in these English sites. For instance, the popular Chinese name "`zhangwei`" appears in both Rockyou and Yahoo. We also note that English names are also widely used in English passwords, yet full names are less popular than last names and first names. As far as we know, for the first time we have explored the name patterns in a large-scale empirical password study.

Equally surprisingly, we find that, on average, 16.99% of Chinese users simply insert a six-digit birthday into their passwords. Besides, about 30.89% of Chinese users employ a $4^+$-digit date as their password building blocks, which is 3.59 times higher than that of English users (i.e. 8.61%); there are 13.49% of Chinese users inserting a four-digit year into their passwords, which is about 3.55 times higher than that of English users (3.80%, which is comparable to the results reported in [38]). We note that there might be some overestimates, for there is no way to tell apart whether some digit sequences are dates or not, e.g., `010101` and `520520`.

TABLE 5
The prevalence of various dictionary words in user passwords (The percentage is taken by dividing the total accounts.)

| Dictionary | Tianya | 7k7k | Dodonew | 178 | CSDN | Duowan | Avg. Chinese | Rockyou | Yahoo | Phpbb | Avg. English |
|---|---|---|---|---|---|---|---|---|---|---|---|
| English_word_lower(len $\geq$ 5) | 2.08% | 2.05% | 3.69% | 0.83% | 3.41% | 2.37% | 2.41% | **23.54%** | **29.49%** | **24.60%** | **25.88%** |
| English_firstname(len $\geq$ 5) | 1.11% | 0.93% | 2.23% | 0.53% | 1.47% | 1.19% | 1.24% | 18.80% | 15.21% | 9.20% | 14.40% |
| English_lastname(len $\geq$ 5) | 2.16% | 2.34% | 4.48% | 1.93% | 3.65% | 2.77% | 2.89% | 20.16% | 20.82% | 15.22% | 18.73% |
| English_fullname(len $\geq$ 5) | 4.03% | 4.30% | 6.14% | 4.99% | 6.58% | 5.07% | 5.18% | 13.05% | 11.35% | 8.25% | 10.88% |
| English_name_any(len $\geq$ 5) | 4.60% | 4.65% | 6.32% | 5.20% | 6.87% | 5.18% | 5.35% | **27.67%** | **26.51%** | **18.71%** | **24.30%** |
| Pinyin_word_lower(len $\geq$ 5) | 7.34% | 8.56% | 10.82% | 10.24% | 11.51% | 9.92% | 9.73% | 3.33% | 2.99% | 2.50% | 2.94% |
| Pinyin_familyname(len $\geq$ 5) | 1.35% | 1.64% | 2.34% | 2.24% | 2.47% | 1.88% | 1.99% | 0.05% | 0.07% | 0.07% | 0.06% |
| Pinyin_fullname(len $\geq$ 5) | **8.39%** | **9.87%** | **12.91%** | **11.81%** | **13.14%** | **11.29%** | **11.24%** | 4.79% | 4.17% | 3.35% | **4.10%** |
| Pinyin_name_any(len $\geq$ 5) | 8.56% | 10.05% | 13.31% | 12.11% | 13.46% | 11.53% | 11.50% | 4.80% | 4.18% | 3.36% | 4.11% |
| Pinyin_place(len $\geq$ 5) | 1.24% | 1.27% | 1.64% | 1.58% | 2.12% | 1.48% | 1.55% | 0.20% | 0.18% | 0.16% | 0.18% |
| PW_with_a_$5^+$-letter_substring | **18.51%** | **19.99%** | **26.95%** | **19.38%** | **28.03%** | **21.70%** | **22.42%** | **71.69%** | **75.93%** | **68.66%** | **72.09%** |
| Date_YYYY | 14.38% | 12.82% | 12.45% | 10.06% | 16.91% | 14.33% | 13.49% | 4.34% | 4.30% | 2.77% | 3.80% |
| Date_YYYYMMDD | 6.06% | 5.42% | 3.93% | 3.94% | 8.78% | 6.17% | 5.72% | 0.10% | 0.05% | 0.09% | 0.08% |
| Date_MMDD | 24.99% | 19.97% | 17.08% | 16.46% | 24.45% | 22.59% | 20.92% | 7.53% | 4.46% | 3.59% | 5.20% |
| Date_YYMMDD | **21.29%** | **15.89%** | **12.70%** | **13.09%** | **20.67%** | **18.28%** | **16.99%** | 3.24% | 1.23% | 1.55% | 2.01% |
| Date_any_above | **36.61%** | **30.39%** | **26.66%** | **27.07%** | **35.30%** | **33.58%** | **31.60%** | 11.33% | 8.77% | 6.45% | 8.85% |
| PW_with_a_digit | 89.49% | 88.42% | 88.52% | 90.76% | 87.10% | 89.26% | 88.93% | 54.04% | 64.74% | 46.14% | 54.97% |
| PW_with_a_$4^+$-digit_substring | 81.64% | 76.98% | 71.90% | 78.76% | 78.38% | 80.60% | 78.04% | 24.72% | 21.85% | 19.33% | 21.97% |
| PW_with_a_$6^+$-digit_substring | **75.59%** | **68.32%** | **61.16%** | **70.02%** | **69.87%** | **73.10%** | **69.68%** | 17.77% | 8.48% | 11.28% | 12.51% |
| PW_with_a_$8^+$-digit_substring | 28.04% | 27.56% | 26.53% | 26.37% | 49.73% | 31.03% | 31.54% | 6.88% | 2.50% | 3.73% | 4.37% |
| Mobile_Phone_Number(11-digit) | **2.90%** | **1.76%** | **2.63%** | **3.97%** | **3.75%** | **2.44%** | **2.91%** | 0.07% | 0.01% | 0.02% | 0.03% |
| PW_with_a_$11^+$-digit_substring | **4.71%** | **2.09%** | **3.39%** | **5.08%** | **7.57%** | **3.35%** | **4.36%** | 0.75% | 0.17% | 0.18% | 0.37% |

These two sequences may be dates, yet they are also likely to be of other semantic meanings (e.g., `520520` sounds like "I love you$\cdots$"). Nevertheless, it doesn't affect our conclusion that birthdays play a vital role in Chinese user passwords.

Another interesting observation is that, about 3% Chinese users just use their 11-digit mobile numbers as passwords, making up 39.59% of all passwords with a $11^+$-digit substring. While there are few passwords longer than 10, if an attacker can determine that the victim uses a long password, she is likely to succeed by just trying the victim's 11-digit mobile number. This reveals a practical attacking strategy against long Chinese passwords.

Note that there are some un-avoidable ambiguities when determining whether a text/digit sequence belongs to a specific dictionary, and improper resolution of these ambiguities would lead to an overestimation or underestimation of human choices. Here we take the dictionary "YYMMDD" for illustration. For example, both `111111` and `520521` fall into "YYMMDD" and are excessively popular, yet it is more likely that users choose them simply because they are easily memorable repetition numbers or meaningful strings, and considering them as ordinary dates would lead to an *overestimation*. Nevertheless, they can really be dates (e.g., `111111` stands for "Jan. 1th, 2011" and `520521` stands for "May 21th, 1952"), and completely excluding them from "YYMMDD" would result in an *underestimation* of the usages of dates. Thus, assuming that user birthdays are randomly distributed, we assign the expectation of frequency of dates (denoted by $E$), instead of zero, to the frequency of these abnormal dates. We manually identify 17 abnormal dates each of which is originally with a frequency greater than $10E$ and appears in every top-1000 list of the six Chinese datasets. In this way, the dilemma can be largely resolved. We similarly tackle 21 abnormal items in the dictionary "MMDD". As for the other 19 dictionaries in Table 5, few abnormal items can be identified, and thus they are processed as usual.

We conjecture that, the two characteristics (i.e., excessively high usages of long Pinyin names and birthdays) of Chinese web passwords would pose a potential for an attacker to greatly reduce her search space, for birthdays and popular names generally are drawn from a much smaller space than random strings should be. If this is the case, then it is fair to say that these two characteristics are just two serious weaknesses of Chinese passwords. In the following, we will establish this conjecture by a series of experiments.

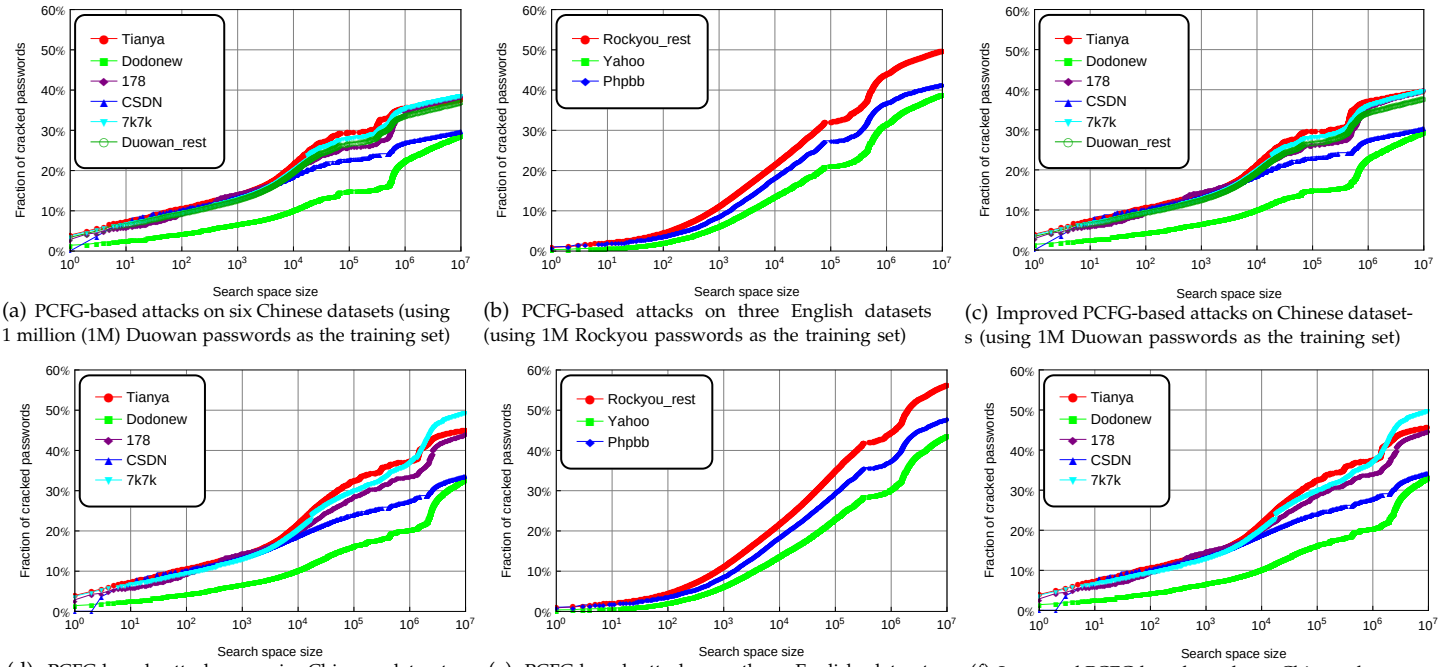
## 4 Strength of Chinese passwords

In this section, we employ the state-of-the-art password attacking algorithms (i.e., PCFG-based and Markov-Chain-based [8]) to evaluate the strength of Chinese passwords, for the traditional entropy-based metric has been demonstrated far from accurate [28]. We further investigate whether the characteristics identified in Sec. 3.3 can be practically exploited to reduce password space and facilitate guessing. We note that probability-threshold graphs can provide cracking results on the full spectrum of passwords, yet their theoretical basis is left as "interesting future research" [8]. Moreover, they only approximate the likelihood of passwords and thus cannot yield precise results. For these reasons, as did in [9], [28], [38], we herein employ guess-number graphs.

### 4.1 PCFG-based attacks

PCFG-based model was first introduced by Weir et al. [12], and it has been revealed to be one of state-of-the-art password cracking algorithms by recent research (e.g., [8], [10]). Unlike JTR [26] that uses ad hoc mangling rules, PCFG-based approach learns the way users create their passwords and automatically derives mangling rules from a training set. Firstly, it divides all the passwords in a training set into segments of similar character sequences and obtains the corresponding base structures and their associated probabilities of occurrence. For example, password "`wanglei@123`" is divided into the $\mathsf{L}$ segment "wanglei", $\mathsf{S}$ segment "@" and $\mathsf{D}$ segment "123", and its base structure is $\mathsf{L}_7\mathsf{S}_1\mathsf{D}_3$. The probability of $\mathsf{L}_7\mathsf{S}_1\mathsf{D}_3$ is $\frac{\#\text{of } \mathsf{L}_7\mathsf{S}_1\mathsf{D}_3}{\#\text{of base structures}}$. Such information is used to generate the probabilistic context-free grammar.

Then, one can derive password guesses in decreasing order of probability, where the probability of any guess is the product of the probabilities of the productions used in its derivation. For instance, the probability of "`liwei@123`" is computed as $P$("`liwei@123`")= $P(\mathsf{L}_5\mathsf{S}_1\mathsf{D}_3)\cdot P(\mathsf{L}_5 \rightarrow \text{liwei})\cdot P(\mathsf{S}_1 \rightarrow \texttt{@})\cdot P(\mathsf{D}_3 \rightarrow 123)$. In Weir et al.'s proposal [12], the




(a) PCFG-based attacks on six Chinese datasets (using 1 million (1M) Duowan passwords as the training set)

(b) PCFG-based attacks on three English datasets (using 1M Rockyou passwords as the training set)

(c) Improved PCFG-based attacks on Chinese datasets (using 1M Duowan passwords as the training set)

(d) PCFG-based attacks on six Chinese datasets (using *all* the Duowan passwords as the training set)

(e) PCFG-based attacks on three English datasets (using 4M Rockyou passwords as the training set)

(f) Improved PCFG-based attacks on Chinese datasets (using *all* the Duowan passwords as the training set)

Fig. 4. General and improved PCFG-based attacks on different groups of datasets

probabilities for digit and symbol segments are learned from the training set by counting, yet letter segments are handled either by also learning from the training set or by using an external input dictionary.

According to Ma et al.'s recent work [8] and on the basis of our past cracking experience, we observe that PCFG-based attack using a letter-segment dictionary, which is directly learned from the training set, to instantiate the letter segments of guesses generally performs better than using an external input dictionary. This is largely due to the fact that until now there has been no effective method to measure the quality of an external input dictionary, while an inappropriate input dictionary would greatly reduce the accuracy of guess generations. What's more, using heuristic approaches to choose the input dictionary may defeat the purpose of using PCFG to avoid heuristic approaches when generating guesses in the first place. As a result, we prefer to instantiate the PCFG letter segments of password guesses by using a dictionary directly learned from the training set.

We divide the nine datasets into two groups according to the languages and user locations. For the Chinese group of test sets, we randomly select 1 million passwords from the Duowan dataset [32] as the PCFG training set (denoted by "Duowan_1M"); For the English group of test sets, we similarly select 1M passwords from Rockyou [30] as the PCFG training set. The rationale underlying our choices of training sets is that, passwords in Duowan and Rockyou exhibit more composition varieties than that of other datasets in the same group, which can be seen from Table 3 to 5. Since we have only used part of Duowan and Rockyou, their remaining passwords as well the other seven datasets are used as the test sets. That being said, no external input dictionary is needed in our general PCFG-based attacks. The attacking results about the Chinese group and English group are depicted in Fig. 4(a) and Fig. 4(b), respectively.

From these two figures we can see that, when the guess number (i.e., search space size) allowed is below about 3,000, Chinese web passwords are generally much *weaker* than English web passwords in terms of the same application domain (i.e., Tianya vs. Rockyou, Dodonew vs. Yahoo, and CSDN vs. phpbb). For example, at 100 guesses, the success rate against Tianya, Dodonew and CSDN is 10.2%, 4.3% and 9.7%, respectively, while their English counterparts are 4.6%, 1.9% and 3.7%, respectively. However, when the search space size is above 10,000, Chinese web passwords are generally much *stronger* than their English counterparts. For example, at 10 million guesses, the success rate against Tianya, Dodonew and CSDN is 37.5%, 28.8% and 29.9%, respectively, while their English counterparts are 49.7%, 39.0% and 41.4%, respectively. The strength gap between these two groups of datasets will be even wider when the guess number further increases. This reveals that a *reversal* has occurred: Chinese passwords are more vulnerable to online guessing attacks (i.e., when the guess number allowed is small), especially the trawling attacks, while English passwords are more prone to offline guessing attacks in which the attacker is not subject to the restriction of the guess number. This "reversal principle" well reconciles two obviously conflicting claims (see Section 1.1) about the security strength of Chinese web passwords.

We observe that, the original PCFG-based algorithm [8], [12] inherently gives extremely low probabilities to password guesses (e.g., "1q2w3e4r" and "1a2b3c4d") that are of a monotonically long structure (e.g., $\mathsf{D}_1\mathsf{L}_1\mathsf{D}_1\mathsf{L}_1\mathsf{D}_1\mathsf{L}_1\mathsf{D}_1\mathsf{L}_1$, or $(\mathsf{D}_1\mathsf{L}_1)_4$ for short). For example, $P(\text{"1q2w3e4r"}) = P((\mathsf{D}_1\mathsf{L}_1)_4) \cdot P(\mathsf{D}_1 \rightarrow 1) \cdot P(\mathsf{L}_1 \rightarrow \mathtt{q}) \cdot P(\mathsf{D}_1 \rightarrow 2) \cdot P(\mathsf{L}_1 \rightarrow \mathtt{w}) \cdot P(\mathsf{D}_1 \rightarrow 3) \cdot P(\mathsf{L}_1 \rightarrow \mathtt{e}) \cdot P(\mathsf{D}_1 \rightarrow 4) \cdot P(\mathsf{L}_1 \rightarrow \mathtt{r})$ can hardly be larger than $10^{-9}$, for it is a multiplication of nine probabilities. As a result, some guesses (e.g., "1q2w3e4r") will never appear in the top $10^7$ guesses generated by the original PCFG-based algorithm, even though they are popular (e.g,

**Algorithm 1:** Improved PCFG-based guesses generation

**Input**: A training set $\mathcal{S}$; A name list *nameList*; A date list *dateList*; A parameter $k$ indicating the desired size of the password guess list that will be generated (e.g., $k = 10^7$)
**Output**: A password guess list $L$ with the $k$ highest ranked items

```
Training (with special attention to monotonically long passwords):
    for password ∈ S do
        for segment ∈ splitToSegments(password) do
            segmentSet.insert(segment)
        baseStructure ← getBaseStructure(password)
        if monotonicallyLong(baseStructure) then
            transformStructureSet.insert(baseStructure)
            baseStructure ← convertToshort(baseStructure)
        baseStructureSet.insert(baseStructure)
        trainingSet.insert(password)
Append name and date lists to the learned segment list
    for name ∈ nameList do
        correctedCount = totalOverlapNameInSegmentSet *
        nameList.getCount(name)/totalOverlapNameInNameList
        if name ∉ segmentSet and correctedCount ≥ 1 then
            segmentSet.insert(name, correctedCount)
    for date ∈ dateList do
        if date ∉ segementSet then
            segementSet.insert(date)
Produce k guesses:
    function guess.calculateProbability()
        guess.probability ←
        baseStructureSet.getProbability(guess.baseStructure)
        if monotonicallyLong(guess.baseStructure) then
            baseStructure ← guess.baseStructure
            guess.probability ← guess.probability *
            transformStructureSet.getProbability(baseStructure)
            guess.baseStructure ←
            convertToShort(baseStructure)
        for segment ∈ splitToSegments(guess.password) do
            guess.probability ← guess.probability *
            segmentSet.getProbability(segment)
    Initialize heap:
        for baseStructure ∈ baseStructureSet do
            for segmentType ∈ baseStructure.segmentTypeSet do
                guess.password +=
                segmentSet.getFirstSegment(segmentType)
            guess.calculateProbability()
            guess.segmentChangedPosition ← 1
            heap.insert(guess)
    while guessCount ≤ k do
        guess ← heap.pop()
        L.insert(guess)
        guessCount++
        for i ← guess.segmentChangedPosition to
        guess.baseStructure.length do
            guessNew.password ←
            guess.changeToNextHightestSegment(i)
            if guessNew.password = null then
                continue
            guessNew.segmentChangedPosition ← i
            guessNew.caluculateProbability()
            heap.insert(guessNew)
```

TABLE 6
Changes caused to the probabilistic context-free grammars

| Training set | Base structures | L segments | D segments | S segments |
|---|---|---|---|---|
| Duowan_1M | 8905+0 | 155693+24416 | 465157+20341 | 865+0 |
| Duowan_All | 20961+0 | 559017+98654 | 1824404+9744 | 2417+0 |

"1q2w3e4r" appears in the top-200 list of every dataset). The essential reason is that the PCFG-based algorithm simply assumes that each segment in a structure is independent. Unfortunately, in many situations this is *not* the case. For instance, the four $\mathsf{D}_1$ segments and $\mathsf{L}_1$ segments in the structure $(\mathsf{D}_1\mathsf{L}_1)_4$ of password "1q2w3e4r" are evidently interrelated with each other.

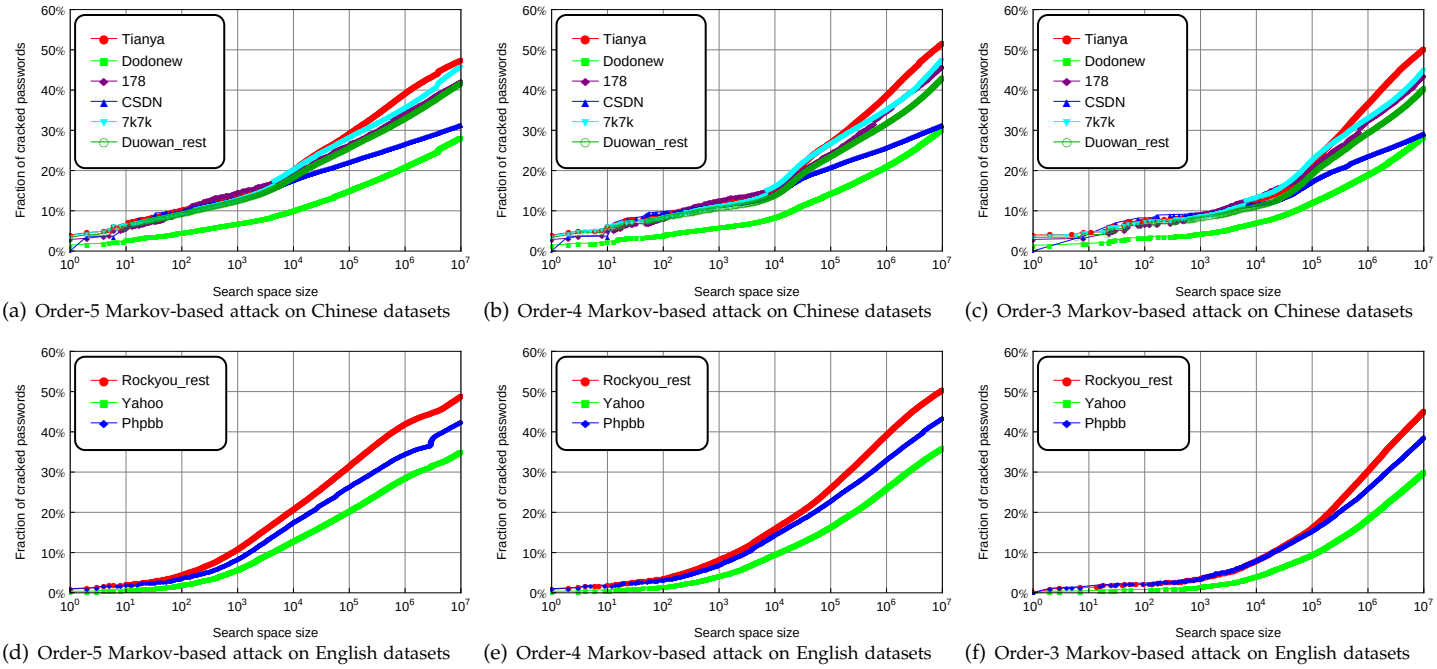
To address this problem, we specially tackle a few structures that are long but simple alternations of short segments by treating them as short structures, e.g., $(\mathsf{D}_1\mathsf{L}_1)_4$ and $(\mathsf{D}_1\mathsf{L}_2)_3$ are converted to $\mathsf{D}_4\mathsf{L}_4$ and $\mathsf{D}_3\mathsf{L}_6$, respectively. In this way, the probability of "1q2w3e4r" now is computed as $P(\text{"1q2w3e4r"}) = P((\mathsf{D}_1\mathsf{L}_1)_4) \cdot P((\mathsf{D}_1\mathsf{L}_1)_4 \rightarrow \mathsf{D}_4\mathsf{L}_4) \cdot P(\mathsf{D}_4 \rightarrow \texttt{1234}) \cdot P(\mathsf{L}_4 \rightarrow \texttt{qwer})$. Note that, our approach is language-irrelevant and constitutes a *general* improvement over the state-of-the-art PCFG-based algorithm [8].

To further exploit the characteristics of Chinese web passwords, we insert the "Pinyin_name_any" dictionary and the six-digit date dictionary (see Section 3.3) into the L-segment dictionary and D-segment dictionary that are learned from the 1 million Duowan passwords using PCFG, respectively. Note that each segment in the L- and D-segment dictionaries is associated with a frequency. As there may have already been some Pinyin names in the original L-segment dictionary and the total frequency of these names (denoted by $n_1$) largely reflects the tendency that users insert Pinyin names into their passwords, we insert a name (its frequency denoted by $f_r$) from the "Pinyin_name_any" dictionary into the L-segment dictionary only if: (1) it is not in the original L-segment dictionary and (2) $\frac{n_1 \cdot f_r}{n_2} \geq 1$, where $n_2$ is the total frequency of names that falls into the intersection of the "Pinyin_name_any" dictionary and the L-segment dictionary. In this way, we manage to only insert a few most frequent ones from an ocean of 2.4M unique names of our name dictionary. On the other hand, as there are only 27.2K items in our six-digit date dictionary, we take all of them into account. More specifically, for any six-digit date that is not in the original D-segment dictionary, we first associate it with a frequency 1 and then insert it into the D-segment dictionary. The resulting changes to the PCFGs learned from the training sets are summarized in Table 6.

Our improved algorithm for the generation of password guesses is illustrated as Algorithm 1. As shown in Fig. 4(c), when the guess number is small (e.g., $10^3$), our improved attack exhibits little improvement in success rate; while the guess number grows, the improvement increases substantially. For example, at $10^5$ guesses, there is 0.09%∼0.85% improvement in success rate; at 1 million guesses, this figure is 1.32∼4.32%; at 10 million guesses, this figure reaches 1.70%∼4.29%. This indicates that, the excessively high usages of Pinyin names and birthdays facilitate an attacker to reduce the search space, and this vulnerability is especially serious when large guesses are allowed.

In 2014, Li et al. [17] reported that using 2 million Dodonew passwords as the training set and at 10 billion guesses, their best cracking record is about 17.30% (see Fig. 5 of [17]). However, our improved attack, which uses only 1 million passwords as the training set and at merely 10 million guesses, is able to achieve success rate from 29.41% to 39.47%. This means that our improved attack can crack 70% to 128% more passwords than Li et al.'s best record. Alarmingly high success rates highlight the urgency of developing effective countermeasures (e.g., more practical password creation policies and more accurate password meters) to alleviate the situation.

Since the effectiveness of PCFG-based attacks depends on the size of the training set, in the following we increase the size of each training set used in the above three experiments, and attempt to see whether the observations made above




(a) Order-5 Markov-based attack on Chinese datasets (b) Order-4 Markov-based attack on Chinese datasets (c) Order-3 Markov-based attack on Chinese datasets

(d) Order-5 Markov-based attack on English datasets (e) Order-4 Markov-based attack on English datasets (f) Order-3 Markov-based attack on English datasets

Fig. 5. Markov-chain-based attacks on different groups of datasets (using *Laplace Smoothing* and *End-Symbol Normalization*). Attacks (a)∼(c) use 1 million Duowan passwords as the training set, while attacks (d)∼(f) use 1 million Rockyou passwords as the training set.

still hold. As shown in Fig. 4(d), at 10M guesses, there is an increase in success rate from 3.89% to 10.78% (the avg. is 6.34%) when we increase the training set (i.e., Duowan) from 1M to 4.98M. Similarly, Fig. 4(e) shows that there is an increase in success rate from 4.67% to 6.44% (the avg. is 5.84%) when we quadruple the training set (i.e., Rockyou). As for our improved PCFG-based attack, Fig. 4(f) shows that at 10M guesses, the success rate is from 33.20% to 49.86% when the training set size reaches 4.98M, which means an increase from 3.79% to 10.39% (the avg. is 5.90%). This suggests that, our improved attack with much less guesses is able to crack 92% to 188% more passwords than Li et al.'s best record (i.e., 17.3%, see Fig. 5 of [17]). Remarkably, the "reversal principle" still holds in this series of experiments.

In our improved PCFG-based attacks, external name segments are added into the PCFG L-segment dictionary during training, and we get gladsome increases in success rates over general PCFG-based attacks (see Figs. 4(c) and 4(f)). However, such improvements are still not so prominent as compared to the prevalence of names in Chinese passwords. To explicate this paradox, we scrutinize the internal process of PCFG-based guess generation and manage to identify its crux. Here we take the improved PCFG-based attack against Tianya (using Duowan as the training set) as an example. During training, we have added 98K name segments (see Table 6) into the L-segment dictionary.

However, as shown in Fig. 6, these 98K name segments only cover 2.88% of the total L segments of the Tianya test set, while the original L segments trained from Duowan can cover 3.77(=13.75/2.88-1) times more of the name segments and 60.59% of the non-name L segments in the Tianya test set. This suggests that the training set Duowan is able to *well* cover the name segments in the test set Tianya, and thus the addition of some extra names would be of limited yields. This observation also holds for the other eight test sets and the detailed results are presented in the supplemental data.

However, this observation does *not* contradict our findings that Pinyin names are prevalent in Chinese passwords and actually, it *does* suggest that when the training set is selected properly, the name segments in passwords can be well guessed. Still, when there is no proper training set available, our improved attack would show its advantages. Moreover, although our improved PCFG-based algorithm might not be optimal, its cracking results represent a new benchmark that any future algorithm should aim to decisively clear.

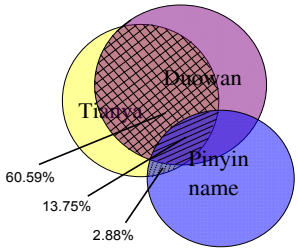



Fig. 6. Coverage of L-segments in the test set Tianya (using Duowan as the training set and Pinyin_name as an extra input dictionary when performing PCFG-based guess generation)

### 4.2 Markov-Chain-based attacks

In our Markov-Chain-based password cracking experiments, as recommended in [8], we consider two smoothing techniques (i.e., Laplace Smoothing and Good-Turing Smoothing) to deal with the data sparsity problem, two normalization techniques (i.e., distribution-based and end-symbol-based) to deal with the unbalanced length distribution problem of passwords. This brings about four attacking scenarios as listed in Table 7. In each scenario we consider three types of markov order (i.e., order-5, 4 and 3) to investigate which order performs best. We note that the other scenario (i.e., backoff with end-symbol normalization) performs "slightly better" than the aforementioned four scenarios, yet it is

"approximately 11 times slower, both for guess generation and for probability estimation" [8]. Therefore, attackers, who particularly care about the cost-effectiveness, are highly unlikely to exploit this scenario. Consequently, we mainly consider the four attacking scenarios as listed in Table 7. Due to space constraints, the detailed password guess generation procedure for Scenario 1 is referred to Algorithm 1 in the supplemental material, and the generation procedures for the other scenarios are quite similar.

TABLE 7
Four Markov-based attacking scenarios

| Attacking scenario | Smoothing | Normalization | Markov order |
|---|---|---|---|
| #1 | Laplace | End-symbol | 3/4/5 |
| #2 | Laplace | Distribution | 3/4/5 |
| #3 | Good-Turing | End-symbol | 3/4/5 |
| #4 | Good-Turing | Distribution | 3/4/5 |

As with PCFG-based attacks, in our implementation we use a max-heap to store the interim results to maintain efficiency. To produce $k = 10^7$ guesses, we employ the strategy of first setting a lower bound (i.e., $10^{-9}$) for the probability of guesses generated, then sorting all the guesses and finally selecting the top $k$ ones. In this way, we are able to reduce the time overheads by 170% at the cost of about 110% increase in storage overheads, as compared to the strategy of producing exactly $k$ guesses. In Laplace Smoothing, it is required to add $\delta$ to the count of each substring and we set $\delta = 0.01$ as suggested by Ma et al. [8]. The cracking results for Scenario 1 are included in Fig. 5. Due to space constraints, the experiments for scenarios 2, 3 and 4 are illustrated in the supplemental material.

There is a subtlety to be noted when implementing the Good-Turing (GT) smoothing technique. We denote $f$ to be the frequency of a password, and $N_f$ to be the frequency of frequency $f$. According to the basic GT smoothing formula, the probability of a string "$c_1c_2\cdots c_l$" in a Markov model of order $n$ is denoted by

$$P(\text{“}c_1c_2\cdots c_{l-1}c_l\text{”}) = \prod_{i=1}^{l} P(\text{“}c_i|c_{i-n}c_{i-(n-1)}\cdots c_{i-1}\text{”}), \quad (1)$$

where the individual probabilities in the product are computed empirically by using the training sets. More specifically, each empirical probability is given by

$$P(\text{“}c_i|c_{i-n}\cdots c_{i-1}\text{”}) = \frac{S(count(c_{i-n}\cdots c_{i-1}c_i))}{\sum_{c\in\Sigma} S(count(c_{i-n}\cdots c_{i-1}c))}, \quad (2)$$

where the alphabet $\Sigma$ includes 95 printable characters on the keyboard plus one special end-symbol (i.e., $c_E$) that denotes the end of a password, and $S(\cdot)$ is defined as:

$$S(f) = (f+1)\frac{N_{f+1}}{N_f}. \quad (3)$$

It can be confirmed that this kind of smoothing works well when $f$ is small, yet it fails for passwords with a high frequency because the estimates for $S(f)$ are not smooth. For instance, `12345` is the most popular 5-character string in Rockyou, occurring $f = 490,044$ times. As there is no 5-character string that occurs 490,045 times and $N_{490045}$ will be zero, implying the basic GT estimator will give a probability 0 for $P$("`12345`"). There have been various improvements suggested in linguistics to cope with this problem, among which is Gale and Hill's "simple Good-Turing smoothing" [39]. This improvement (denoted by SGT) is famous for its simplicity and accuracy, and we adopt it in this work. SGT takes two steps of smoothing, and the details can be found in the supplemental material.

In 2014, Ma et al. [8] introduced GT smoothing into Markov-based attacks to facilitate more accurate generation of password guesses, yet little attention has been paid to the unsoundness of GT for high frequency events as illustrated above. To the best of our knowledge, we for the first time well explicate the combination uses of GT and SGT in Markov-based password cracking.

From the experiments we observe that for both Chinese and English test sets: (1) At large guesses (i.e., no less than $2*10^6$), order-4 markov-chain evidently performs better than the other two orders, while at small guesses (i.e., less than $10^6$) the larger the order, the better the performance will be; (2) There is not much difference in performance between Laplace and Good-Turing Smoothing at small guesses, while the advantage of Laplace Smoothing gets greater as the guess number increases; (3) End-symbol-based normalization always performs better than the distribution-based approach, while at small guesses its advantages will be more obvious. This suggests that at large guesses, the attacks preferring order-4, Laplace Smoothing and end-symbol-based normalization (see Fig. 5(b) and Fig. 5(e)) perform the best among all the series of Markov-chain-based attacks; at small guesses (e.g., less than $10^6$), the attacks preferring order-5, Laplace Smoothing and end-symbol-based normalization (see Fig. 5(a) and Fig. 5(d)) perform best.

It is worth noting that, the "reversal principle" also applies in all the Markov-based experiments. For example, in order-4 Markov-based experiments (see Figs. 5(b) and 5(e)), we can see that, when the guess number is below about 7000, Chinese passwords are generally much *weaker* than their English counterparts. For example, at 1000 guesses, the success rate against Tianya, Dodonew and CSDN is 11.8%, 6.3% and 11.6%, respectively, while their English counterparts (i.e., Rockyou, Yahoo and Phpbb) is merely 8.1%, 4.3% and 7.1%, respectively. However, when the guess number allowed is over $10^4$, Chinese passwords are generally *stronger* than their English counterparts. For example, at 1 million guesses, the success rate against Tianya, Dodonew and CSDN is 38.7%, 21.2% and 25.9%, respectively, while their English counterparts is 39.2%, 26.1% and 33.3%, respectively.

## 5 Some Critical Implications

We now discuss some important implications that our findings revealed in previous sections are highly likely to carry.

### 5.1 Implications for password cracking

Password cracking algorithms are not only necessary tools for security administrators to obtain a realistic picture of the security provided by a user-generated password, but also they can be used to facilitate information forensics (e.g., for law enforcement agencies to recover encrypted data). Our results have three main implications for password cracking. Firstly, our findings in Sec. 3 show that Chinese passwords have vastly different letter distribution, structure and semantic patterns as compared to English passwords, and thus it is crucial for cracking algorithms to be trained on relevant Chinese datasets when targeting Chinese passwords.

Secondly, when using PCFG-based attacks, it is better to *mainly* employ the L-segments directly learned from the training set and may *additionally* include some external special dictionaries to instantiate the L-segments of password guesses. This implication accords with the findings in [8]. Its validity can be well established by the fact that, given the same guess numbers and against the same test sets, our PCFG-based attacks can obtain much higher success rates (see Section 4.1) than those of the PCFG-based attacks suggested in [12], [17] where external dictionaries were *mainly* used to instantiate the L-segments of password guesses.

Thirdly, compared to Markov-based attacks, PCFG-based ones are simpler to implement (in terms of both computation and memory cost), and they perform equally well (even better, see Figs. 4 and 5) when the guess number is small (e.g., a few thousands). For large guess numbers, order-4 Markov-based attacks are the best choices. Note that we have only shown the Markov-based cracking results when the guess number is below $10^7$, there is a potential that order-3 Markov-based attacks will outperform order-4 and 5 ones at larger guess numbers (e.g., $10^{14}$).

### 5.2 Implications for password strength meters

In Section 1.1 we have shown that the password strength meters (PSMs) of four Internet-scale service providers are highly inconsistent at assessing the security of (weak) Chinese passwords. Failing to provide coherent feedback on user password choices would have great negative effects such as user confusion, frustration and distrust. Carnavalet and Mannan [14] suggested that PSMs "can simplify challenges by limiting their primary goal only to detecting *weak* passwords, instead of trying to distinguish a good, very good, or great password." It follows that the essential step of a PSM would be to identify the characteristics of weak passwords. From our findings in Section 3.3 and Section 4.1, it is evident that for passwords of Chinese users, the incorporation of long Pinyin words or full/family names is adequate evidence for a "weak" decision. Other signs of weak passwords are the incorporation of birthdates and simple patterns like repetition and palindrome.

The "reverse principle" revealed in Section 4 shows that Chinese passwords are more vulnerable to online guessing attacks, which is highly due to the fact that Chinese passwords are more concentrated (i.e., some passwords are overly popular, see Table 3). Thus, a special blacklist that includes a moderate number (e.g., 50K as suggested in [28]) of most common Chinese passwords (e.g., learned from various leaked Chinese datasets) would be highly helpful for Chinese users to avoid trivial online guessing attacks. Any password falling into this list shall be deemed weak. However, it is well known that if some popular passwords are banned, new popular ones will arise. These new popular passwords may be complex and subtle to detect. Hence, whenever possible, PSMs shall further employ state-of-the-art cracking algorithms with consideration of local password characteristics (e.g., service and language) to assess the real strength that user-generated passwords can provide.

### 5.3 Implications for password creation policies

Password creation policies are generally used along with PSMs to nudge users towards better passwords. While password creation policies tell a user what constitutes a good (or an acceptable) password, PSMs feedback to a user how her submitted password performs (weak or not?). It is interesting to see that CSDN enforces a minimum length-8 policy (as shown in Fig. 2 and [34], 97.83% passwords in CSDN are of length $8^+$), while Dodonew enforces no apparent rule (i.e., neither minimum length nor character set requirement) as is evident from Table 3. However, Figs. 4 and 5 indicate that, given any guess number below $10^7$, passwords from CSDN is significantly weaker than passwords from Dodonew. A plausible reason is that Dodonew provides e-commerce services, and most users perceive it as important. As a result, users *rationally* choose more complex passwords for it.

In 2012, Bonneau [7] cast doubt on the hypothesis that users will rationally select more secure passwords to protect their more important accounts. However, in 2013 Egelman et al. [15] initiated a field study involving 51 students and confirmed this hypothesis. In 2014, Stobert and Biddle [18] interviewed 27 participants to investigate user behaviour in managing passwords, and their results also corroborate this hypothesis. As far as we know, here we for the first time provide a large-scale *empirical evidence* (i.e., on the basis of 6.43 million CSDN passwords and 16.26 million Dodonew passwords) that supports for this hypothesis.

We also note that though the overall security of Dodonew passwords are higher than passwords from the five other Chinese sites, many popular passwords dwelling in Dodonew also appear in other less sensitive sites (see Table 3 for a concrete example). This might be due to that users inadvertently choose popular passwords and that many users reuse the same password across multiple sites (43%∼51% of users reuse passwords as reported in [38]). What's even more dangerous is that many users fail to recognize different categories of accounts, because achieving this goal is not an easy task [40]. Further considering the "over-constrained nature of authentication" on the Web [41] and the "finite-effort user" [22], we suggest that when designing password policies, instead of merely insisting on stringent rules, security administrators should put more efforts on helping users gain more accurate perceptions of the importance of the accounts to be protected and on guiding users towards better ability of recognizing different categories of accounts. Both efforts are essential for common users to responsibly allocating (i.e., selecting one candidate from their limited pool of passwords memorized [40]) passwords.

## 6 Conclusion

In this paper, we have conducted an extensive empirical study of 130 million real-life passwords, including 100 million from China and 30 million from US, the largest password corpus ever studied. To the best of knowledge, we, for the first time, explore several fundamental properties (e.g., the frequency distribution and the distance between different letter distributions) that characterize user-chosen passwords. By using a comparison approach, we have identified a number of interesting characteristics of Chinese passwords, such as the excessively common usage of long Pinyin names, birthdays and mobile numbers. We further performed a series of experiments by using the state-of-the-art password cracking algorithms as well as our improved PCFG-based algorithm to evaluate password strength. Our results show

that, the identified characteristics can be exploited by an guessing attacker to largely reduce her search space.

Of particular interest is our observation that the "reversal principle" applies: when the guess number allowed is small (e.g., less than $10^4$), Chinese passwords are much weaker than their English counterparts, yet this relationship will be reversed when the guess number is large (e.g., larger than $10^5$). This indicates that Chinese passwords are more susceptible to *online, trawling attacks*, while English ones are more vulnerable to *offline guessing attacks*, which for the first time well reconciles two conflicting claims [7], [17]. Considering the comprehensiveness of our exploration (and the ampleness of our corpus), we believe this work constitutes an important step forward in understanding Chinese passwords and provides substantial "ground truth" for better design of future password policies, meters, etc.

While we have shown that both group of users choose passwords by miraculously following the Zipf's law, the underlying mechanism that gives rise to the emergence of this law is left as an open issue. Another line of interesting future work would be to testify the effectiveness of our methodologies and observations on large-scale passwords from other non-English-speaking populations.

## References


[1] R. Morris and K. Thompson, "Password security: A case history," *Comm. of the ACM*, vol. 22, no. 11, pp. 594–597, 1979.

[2] B. Zhu, J. Yan, G. Bao, M. Mao, and N. Xu, "Captcha as graphical passwords–a new security primitive based on hard AI problems," *IEEE Trans. Inform. Forensics Security*, vol. 9, no. 6, pp. 891–904, 2014.

[3] X. Huang, Y. Xiang, E. Bertino, J. Zhou, and L. Xu, "Robust multi-factor authentication for fragile communications," *IEEE Trans. Depend. Secur. Comput.*, vol. 11, no. 6, pp. 568–581, 2014.

[4] 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*. IEEE, pp. 553–567.

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

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

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

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

[9] R. Veras, C. Collins, and J. Thorpe, "On the semantic patterns of passwords and their security impact," in *Proc. NDSS 2014*, pp. 1–16.

[10] M. L. Mazurek, S. Komanduri, T. Vidas, L. F. Cranor, P. G. Kelley, R. Shay, and B. Ur, "Measuring password guessability for an entire university," in *Proc. CCS 2013*. ACM, Nov. 4–8 2013, pp. 173–186.

[11] 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.

[12] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *Proc. 30th IEEE Symp. on Security and Privacy*. IEEE, 2009, pp. 391–405.

[13] F. Bergadano, B. Crispo, and G. Ruffo, "High dictionary compression for proactive password checking," *ACM Trans. on Information and System Security*, vol. 1, no. 1, pp. 3–25, 1998.

[14] X. Carnavalet and M. Mannan, "From very weak to very strong: Analyzing password-strength meters," in *Proc. NDSS 2014*, pp. 1–16.

[15] 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. CHI 2013*. ACM, pp. 2379–2388.

[16] *CNNIC Released the 35th Statistical Report on Internet Development in China*, CNNIC, Feb. 2015, http://www.apira.org/news.php?id=1732.

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

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

[19] D. V. Klein, "Foiling the cracker: A survey of, and improvements to, password security," in *Proc. of USENIX Security*, 1990, pp. 5–14.

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

[21] I. Erguler, "Achieving flatness: Selecting the honeywords from existing user passwords," *IEEE Trans. Depend. Secur. Comput.*, 2015, doi: 10.1109/TDSC.2015.2406707.

[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 Security 2014*, Aug. 2014, pp. 575–590.

[23] B. L. Riddle, M. S. Miron, and J. A. Semo, "Passwords in use in a university timesharing environment," *Computers & Security*, vol. 8, no. 7, pp. 569–579, 1989.

[24] A. S. Brown, E. Bracken, and S. Zoccoli, "Generating and remembering passwords," *Applied Cogn. Psych.*, vol. 18, no. 6, pp. 641–651, 2004.

[25] R. Veras, J. Thorpe, and C. Collins, "Visualizing semantics in passwords: The role of dates," in *Proc. VizSec 2012*. ACM, pp. 88–95.

[26] S. Designer, *John the Ripper password cracker*, Feb. 1996, http://www.openwall.com/john/.

[27] D. Florencio and C. Herley, "A large-scale study of web password habits," in *Proc. WWW 2007*. ACM, 2007, pp. 657–666.

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

[29] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proc. CCS 2005*. ACM, pp. 364–372.

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

[31] V. Katalov, *Yahoo!, Dropbox and Battle.net Hacked: Stopping the Chain Reaction*, Feb. 2013, http://blog.crackpassword.com/tag/yahoo/.

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

[33] J. Huang, H. Jin, F. Wang, and B. Chen, "Research on keyboard layout for chinese pinyin ime," *Journal Of Chinese Information Processing*, vol. 24, no. 6, pp. 108–113, 2010.

[34] D. Wang, G. Jian, X. Huang, and P. Wang, "Zipf's law in passwords," Cryptology ePrint Archive, Report 2014/631, pp. 1–24, 2014, http://eprint.iacr.org/2014/631.pdf.

[35] R. A. Butler, *List of the Most Common Names in the U.S.*, Jan. 2014, http://names.mongabay.com/most_common_surnames.htm.

[36] J. Goldman, *Chinese Hackers Publish 20 Million Hotel Reservations*, Dec. 2013, http://www.esecurityplanet.com/hackers/chinese-hackers-publish-20-million-hotel-reservations.html.

[37] *Sogou Internet thesaurus*, Sogou Labs, April 17 2014, http://www.sogou.com/labs/dl/w.html.

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

[39] W. Gale and G. Sampson, "Good-turing smoothing without tears," *Journal of Quantitative Linguistics*, vol. 2, no. 3, pp. 217–237, 1995.

[40] R. Nithyanand and R. Johnson, "The password allocation problem: strategies for reusing passwords effectively," in *Proc. WPES 2013*. ACM, 2013, pp. 255–260.

[41] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The past, present, and future of password-based authentication on the web," *Communi. of the ACM*, 2015, in press.



**Ding Wang** received his B.S. Degree in Information Security from Nankai University, China, in 2008. Currently, he is pursuing his Ph.D. degree at Peking University, China. He has published more than 20 refereed research papers at IEEE, Elsevier and Wiley journals, and conferences such as DBSec 2012, ISC 2013 and SecureComm 2014. He was awarded the Top-Ten Distinguished Graduate Academic Star of the University in 2012. His research interests include cryptography and system security.

**Haibo Cheng** received his B.S. and M.S. degrees in Fundamental Mathematics from Nankai University in 2011 and Peking University in 2014, respectively. He is currently a Ph.D. candidate at School of EECS, Peking University, China. His main research interests include applied cryptography, in particular, security data analytics.

**Qianchen Gu** received his B.S. degree in Computer Science from Peking University, China. Currently, he a graduate in the School of EECS, Peking University. His research interests include information security and concurrent computation.

**Ping Wang** received his B.S. degree from University of Electronic Science and Technology of China in 1983, and Ph.D. degree from University of Massachusetts in 1996. Currently, he is a Professor in Peking University. He served as TPC co-chairs of RFIDSec'11 Asia. He was co-recipient of three Best Paper Awards from different professional societies. He has wide interests in system security, system software and distributed computing.

# Understanding Passwords of Chinese Users: Characteristics, Security and Implications (Supplemental File)

**Abstract**—This supplementary file is composed of six sections. Section 1 shows the top-10 most popular password patterns with digits in Chinese web passwords. Section 2 deals with the semantic patterns regarding Pinyin names and dates that are at least 4-characters long. Section 3 illustrates the coverage of name segments in test sets. Section 4 describes the Markov-based algorithm for the generation of password guesses. Section 5 explicates a subtlety about Good-Turing smoothing in the Makov-based password cracking algorithm. Section 6 demonstrates the Markov-based cracking results under three attacking scenarios.

**Index Terms**—Password authentication, Password structure, Semantic pattern, Markov model.

## 1 Top-10 most popular password patterns with digits

As we have seen that digits are popular in top 10 passwords of Chinese datasets, whether are they popular in the whole datasets? To answer this question, we investigate the frequencies of password patterns that involve digits, and the results (on top 10 most frequent patterns) can be found in Table 1. The first row of the table denotes the pattern of a password as in [1] (L denotes a lower-case sequence, D for digit sequence, U for upper-case sequence, and S for symbol sequence). We see that an average of more than 50% of Chinese web passwords are only composed of digits, while this value of English datasets is only 15.77%. In contrast, English users prefer the pattern LD. Note that, all the percentages hereafter in this work are taken by dividing the corresponding total accounts (e.g., the percentage at the upper-left corner of Table 1 is computed as $\frac{\#\text{of passwords with pattern D}}{\#\text{of total passwords in Tianya}} = \frac{19706174}{30901241} = 63.77\%$).

It is surprising to see that, the sum of merely the first two patterns D and LD accounts for over 70% for *every* Chinese datasets, which indicates that Chinese users excessively employ digits to build their passwords. This may be due in large part to the fact that most Chinese users are unfamiliar with English language (and Roman letters on the keyboard). If this is the case, is there any meaningful information underlying these digit sequences?

## 2 Semantic patterns in user-chosen passwords

To gain an insight into the underlying semantic patterns, we construct several dictionaries of different semantic categories and investigate their prevalence (see Table 2). "English_word_lower" is from http://www.mieliestronk.com/wordlist.html and it contains about 58,000 popular lower-case English words. "English_lastname" is a dictionary consisting of 18,839 last (family) names with over 0.001% frequency in the US population during the 1990 census according to US Census Bureau [2]. "English_firstname" contains 5,494 most common first names (1,219 male and 4,275 female names) in US [2]. "English_fullname" is a cartesian product of "English_firstname" and "English_lastname", containing about 1.04 million most common English full names. To get a Chinese full name dictionary, we make use of the 20 million hotel reservations dataset [3] leaked in Dec. 2013. The Chinese family name dictionary includes 504 family names (see http://en.wikipedia.org/wiki/Hundred_Family_Surnames) which are officially recognized in China. Since the first names of Chinese users are widely distributed and can be almost any combinations of Chinese words, we do not consider them in this work. As the names are originally in Chinese, we transferred them into Pinyin without tones by using a Python procedure from https://pypinyin.readthedocs.org/en/latest/ and removed the duplicates. We call these two name dictionaries as "Pinyin_fullname" and "Pinyin_familyname", respectively.

"Pinyin_word_lower" is a Chinese word dictionary known as "SogouLabDic.dic", and "Pinyin_place" is a Chinese place dictionary. Both of them are from [4] and also originally in Chinese, and we translate them into Pinyin in the same way as we tackle the name dictionaries. "Mobile_number" consists of all Chinese mobile numbers, which are 11-digit strings and the first three digits belong to a small specified collection (e.g., 130, 139, 150, 185 and so on). As for the birthday dictionaries, we use patterns to match digit strings that might be birthdays. For example, "YYYYMMDD" stands for a birthday pattern that the first four digits indicates years (from 1900 to 2014), the middle two represents months (from 01 to 12) and the last two denotes dates (from 01 to 31). "PW with a $l^+$-letter substring" is a subset of the corresponding dataset in each column (see Table 2) and consists of all passwords that include a letter substring *no shorter than* $l$, and similarly for "PW with a $l^+$-digit substring".

Table 2 shows the various semantic patterns existing in Chinese and English web passwords. We can see that, a large fraction of English users tend to use raw English words as their password building blocks. More specially, 43.33% English users insert a 4-letter or longer (denoted by $4^+$-letter) *word* into their passwords, and this figure accounts for more than half of the total passwords with a $4^+$-letter substring; 25.88% English users insert a $5^+$-letter *word* into

TABLE 1
Password patterns with digits (The percentage is taken by dividing the corresponding total accounts.)

| Patterns | Tianya | 7k7k | Dodonew | 178 | CSDN | Duowan | Avg. Chinese | Rockyou | Yahoo | Phpbb | Avg. English |
|---|---|---|---|---|---|---|---|---|---|---|---|
| D | **63.77%** | **59.62%** | **30.76%** | **48.07%** | **45.01%** | **52.84%** | **52.93%** | 15.94% | 5.89% | 12.06% | 15.77% |
| LD | 14.71% | 17.98% | 43.50% | 31.12% | 26.14% | 23.97% | 23.72% | **27.70%** | **38.27%** | **19.14%** | **27.78%** |
| Sum of top2 | **78.48%** | **77.60%** | **74.25%** | **79.20%** | **71.15%** | **76.81%** | **76.66%** | 43.64% | 44.16% | 31.20% | 43.55% |
| DL | 4.12% | 3.91% | 7.55% | 6.25% | 5.88% | 5.83% | 5.25% | 2.54% | 5.31% | 2.03% | 2.57% |
| LDL | 1.17% | 0.89% | 1.46% | 1.27% | 1.64% | 1.52% | 1.24% | 1.62% | 3.30% | 3.64% | 1.66% |
| UD | 0.51% | 0.34% | 1.90% | 0.62% | 1.62% | 0.37% | 0.83% | 1.35% | 0.56% | 0.37% | 1.33% |
| ULD | 0.27% | 0.12% | 0.27% | 0.05% | 0.50% | 0.31% | 0.25% | 0.94% | 2.48% | 1.03% | 0.96% |
| DLD | 0.43% | 0.31% | 0.39% | 0.38% | 0.52% | 0.45% | 0.41% | 0.42% | 0.94% | 0.78% | 0.43% |
| LSD | 0.31% | 0.06% | 0.36% | 0.08% | 0.66% | 0.37% | 0.30% | 0.50% | 0.39% | 0.17% | 0.50% |
| LDLD | 0.28% | 0.22% | 0.28% | 0.47% | 0.47% | 0.34% | 0.31% | 0.42% | 0.97% | 1.02% | 0.43% |
| LDS | 0.23% | 0.07% | 0.11% | 0.10% | 0.54% | 0.51% | 0.21% | 0.21% | 0.26% | 0.07% | 0.21% |
| Sum of top10 | 85.82% | 83.52% | 86.56% | 88.41% | 82.97% | 86.51% | **85.45%** | 51.64% | 58.37% | 40.31% | **51.64%** |

TABLE 2
The prevalence of various dictionary words in user passwords (The percentage is taken by dividing the total accounts.)

| Dictionary | Tianya | 7k7k | Dodonew | 178 | CSDN | Duowan | Avg. Chinese | Rockyou | Yahoo | Phpbb | Avg. English |
|---|---|---|---|---|---|---|---|---|---|---|---|
| English_word_lower(len $\geq 4$) | 5.42% | 5.57% | 9.33% | 4.16% | 9.75% | 6.68% | 6.82% | **41.99%** | **47.55%** | **40.44%** | **43.33%** |
| English_firstname(len $\geq 4$) | 5.15% | 5.01% | 8.51% | 5.09% | 7.68% | 6.16% | 6.27% | 30.19% | 25.78% | 17.51% | 24.49% |
| English_lastname(len $\geq 4$) | 7.52% | 8.23% | 13.25% | 8.25% | 13.32% | 9.69% | 10.04% | 38.01% | 38.96% | 31.15% | 36.04% |
| English_fullname(len $\geq 4$) | 5.80% | 6.16% | 9.04% | 7.10% | 9.31% | 7.36% | 7.46% | 16.72% | 14.59% | 11.32% | 14.21% |
| English_name_any(len $\geq 4$) | 9.00% | 9.46% | 15.42% | 9.48% | 15.02% | 11.23% | 11.60% | **46.24%** | **45.93%** | **36.30%** | **42.82%** |
| Pinyin_word_lower(len $\geq 4$) | 9.18% | 10.67% | 14.61% | 12.51% | 14.20% | 12.55% | 12.29% | 13.61% | 12.01% | 10.56% | 12.06% |
| Pinyin_familyname(len $\geq 4$) | 6.34% | 7.14% | 10.04% | 9.21% | 10.35% | 8.44% | 8.59% | 1.45% | 1.32% | 1.20% | 1.33% |
| Pinyin_fullname(len $\geq 4$) | 9.87% | 11.42% | 15.90% | 13.27% | 15.32% | 13.42% | 13.20% | 15.22% | 13.43% | 11.78% | 13.48% |
| Pinyin_name_any(len $\geq 4$) | **10.91%** | **12.42%** | **18.06%** | **14.92%** | **17.18%** | **14.81%** | **14.72%** | 15.83% | 14.04% | 12.34% | 14.07% |
| Pinyin_place(len $\geq 4$) | 1.95% | 2.27% | 2.87% | 2.50% | 3.33% | 2.62% | 2.59% | 1.27% | 1.03% | 0.89% | 1.06% |
| PW_with_a_$4^+$-letter_substring | **22.03%** | **23.01%** | **32.54%** | **23.73%** | **33.23%** | **25.80%** | **26.72%** | **77.84%** | **84.00%** | **76.92%** | **79.59%** |
| English_word_lower(len $\geq 5$) | 2.08% | 2.05% | 3.69% | 0.83% | 3.41% | 2.37% | 2.41% | **23.54%** | **29.49%** | **24.60%** | **25.88%** |
| English_firstname(len $\geq 5$) | 1.11% | 0.93% | 2.23% | 0.53% | 1.47% | 1.19% | 1.24% | 18.80% | 15.21% | 9.20% | 14.40% |
| English_lastname(len $\geq 5$) | 2.16% | 2.34% | 4.48% | 1.93% | 3.65% | 2.77% | 2.89% | 20.16% | 20.82% | 15.22% | 18.73% |
| English_fullname(len $\geq 5$) | 4.03% | 4.30% | 6.14% | 4.99% | 6.58% | 5.07% | 5.18% | 13.05% | 11.35% | 8.25% | 10.88% |
| English_name_any(len $\geq 5$) | 4.60% | 4.65% | 6.32% | 5.20% | 6.87% | 5.18% | 5.35% | **27.67%** | **26.51%** | **18.71%** | **24.30%** |
| Pinyin_word_lower(len $\geq 5$) | 7.34% | 8.56% | 10.82% | 10.24% | 11.51% | 9.92% | 9.73% | 3.33% | 2.99% | 2.50% | 2.94% |
| Pinyin_familyname(len $\geq 5$) | 1.35% | 1.64% | 2.34% | 2.24% | 2.47% | 1.88% | 1.99% | 0.05% | 0.07% | 0.07% | 0.06% |
| Pinyin_fullname(len $\geq 5$) | **8.39%** | **9.87%** | **12.91%** | **11.81%** | **13.14%** | **11.29%** | **11.24%** | 4.79% | 4.17% | 3.35% | **4.10%** |
| Pinyin_name_any(len $\geq 5$) | 8.56% | 10.05% | 13.31% | 12.11% | 13.46% | 11.53% | 11.50% | 4.80% | 4.18% | 3.36% | 4.11% |
| Pinyin_place(len $\geq 5$) | 1.24% | 1.27% | 1.64% | 1.58% | 2.12% | 1.48% | 1.55% | 0.20% | 0.18% | 0.16% | 0.18% |
| PW_with_a_$5^+$-letter_substring | **18.51%** | **19.99%** | **26.95%** | **19.38%** | **28.03%** | **21.70%** | **22.42%** | **71.69%** | **75.93%** | **68.66%** | **72.09%** |
| Date_YYYY | 14.38% | 12.82% | 12.45% | 10.06% | 16.91% | 14.33% | 13.49% | 4.34% | 4.30% | 2.77% | 3.80% |
| Date_YYYYMMDD | 6.06% | 5.42% | 3.93% | 3.94% | 8.78% | 6.17% | 5.72% | 0.10% | 0.05% | 0.09% | 0.08% |
| Date_MMDD | 24.99% | 19.97% | 17.08% | 16.46% | 24.45% | 22.59% | 20.92% | 7.53% | 4.46% | 3.59% | 5.20% |
| Date_YYMMDD | **21.29%** | **15.89%** | **12.70%** | **13.09%** | **20.67%** | **18.28%** | **16.99%** | 3.24% | 1.23% | 1.55% | 2.01% |
| Date_any_above | **36.61%** | **30.39%** | **26.66%** | **27.07%** | **35.30%** | **33.58%** | **31.60%** | 11.33% | 8.77% | 6.45% | 8.85% |
| PW_with_a_digit | 89.49% | 88.42% | 88.52% | 90.76% | 87.10% | 89.26% | 88.93% | 54.04% | 64.74% | 46.14% | 54.97% |
| PW_with_a_$4^+$-digit_substring | 81.64% | 76.98% | 71.90% | 78.76% | 78.38% | 80.60% | 78.04% | 24.72% | 21.85% | 19.33% | 21.97% |
| PW_with_a_$6^+$-digit_substring | **75.59%** | **68.32%** | **61.16%** | **70.02%** | **69.87%** | **73.10%** | **69.68%** | 17.77% | 8.48% | 11.28% | 12.51% |
| PW_with_a_$8^+$-digit_substring | 28.04% | 27.56% | 26.53% | 26.37% | 49.73% | 31.03% | 31.54% | 6.88% | 2.50% | 3.73% | 4.37% |
| Mobile_Phone_Number(11-digit) | **2.90%** | **1.76%** | **2.63%** | **3.97%** | **3.75%** | **2.44%** | **2.91%** | 0.07% | 0.01% | 0.02% | 0.03% |
| PW_with_a_$11^+$-digit_substring | **4.71%** | **2.09%** | **3.39%** | **5.08%** | **7.57%** | **3.35%** | **4.36%** | 0.75% | 0.17% | 0.18% | 0.37% |

their passwords, and this figure accounts for more than one third of the total passwords with a $5^+$-letter substring. In contrast, few Chinese users choose raw Pinyin words or English words to build passwords, yet they prefer Pinyin names, especially full names. It is surprising to see that, of all the Chinese passwords (22.42%) that include a $5^+$-letter substring, more than half (11.24%) include a $5^+$-letter Pinyin full name, and this tendency is more pronounced when considering Chinese passwords with a $4^+$-letter full name. There is even a non-negligible proportion (i.e., 4.10%) of passwords in English datasets that contains a $5^+$-letter full Pinyin name, and a reasonable explanation for this observation may be that many Chinese users have created accounts in these English sites. We also note that English names are also widely used in English passwords, yet full names are less popular than last names and first names. As far as we know, for first time we have explored the name patterns in a large-scale empirical password study.

Equally surprisingly, we find that, on average, about 20% of Chinese users simply insert a six-digit birthday into their passwords. Besides, about 30.89% of Chinese users employ a $4^+$-digit date as their password building blocks, which is 3.59 times higher than that of English users (i.e. 8.61%); there are 13.49% of Chinese users inserting a four-digit year into their passwords, which is about 3.55 times higher than that of English users (3.80%, which is comparable to results reported in [5]). We note that there might be some overestimates, for there is no way to tell apart whether some digit sequences are dates or not, e.g., `010101` and `520520`. These two sequences may be dates, yet they are also likely to be of other semantic meanings (e.g., `520520` can stand for "I love you I love you"). Nevertheless, it doesn't affect our conclusion that birthdays play a vital part when Chinese users build their passwords. Another interesting observation

is that, about 3% Chinese users just use their 11-digit mobile numbers as passwords, making up 39.59% of all passwords with a $11^+$-digit substring. While there are few passwords longer than 10, if an attacker can determine that the victim uses a long password, she is likely to succeed by just trying the victim's 11-digit mobile number. This reveals a practical attacking strategy against long Chinese passwords.

## 3 Semantic patterns in passwords

As we have shown in Sec.4.1 of the main text, the training set (i.e., Duowan) is able to well cover the name segments in the test set (i.e., Tianya) and thus the addition of some extra names would be of limited yields. This observation also holds for the other eight test sets and the detailed results are summarized in Table 3, where "Duowan1M" is Duowan_1M for short and "PY_name" is Pinyin_name for short. The fraction of L-segments in the test set $y$ that can be covered by the set $x$ is denoted by CoL($x$).

Table 3 shows CoL($x$) is at least 11 times larger than CoL(Pinyin_name)$-$CoL($x$), and CoL(Pinyin_name)$\cap$CoL($x$) is at least 1.9 times larger than CoL(Pinyin_name)$-$CoL($x$), no matter $x$ =Duowan or Duowan_1M. As a result, adding extra names into the PCFG L-segments when training is of limited yields. Note that, this does *not* contradict our observation that Pinyin names are prevalent in Chinese web passwords and actually, this does suggest that when the training set is selected properly, the name segments in passwords can be well covered (guessed). Still, when there is no proper training set available, our improved attack would demonstrate its advantages. Moreover, although our improved PCFG-based algorithm might not be optimal, its cracking results represent a new benchmark that any future algorithm should aim to decisively clear.

## 4 Algorithm for Markov-based guess generation

Markov-Chain-based password cracking model is inspired by the Markov-Chain models widely used in the natural language processing (NLP) domain, and it was first introduced in [6] and later explored in [7] to reduce the password search space, yet no advanced NLP techniques (e.g., smoothing and normalization) has been employed to cope with the data sparsity and overfitting problem. In 2014, Ma et al. [8] investigated these issues and reported that, "when using a Markov-Chain of an order that is high enough, but not too high, and with some ways to deal with overfitting, would perform reasonably well" and in some cases even perform significantly better than the PCFG-based cracking model. Consequently, here we further conduct a series of Markov-Chain-based attacks on the nine real-world password datasets.

In our experiments, as recommended in [8], we consider two smoothing techniques (i.e., Laplace Smoothing and Good-Turing Smoothing) to deal with the data sparsity problem, two normalization techniques (i.e., distribution-based and end-symbol-based) to deal with the unbalanced length distribution problem of passwords. This brings about four attacking scenarios as listed in Table 7 of the main text. In each scenario we consider three types of markov order (i.e., order-5, 4 and 3) to investigate which order performs best. Due to space constraints, here we only illustrate the detailed password guess generation procedure for experiments using Laplace smoothing and end-symbol-based normalization (i.e., Scenario 1) in Algorithm 1, and the generation procedures for the three other scenarios (see Table 7 of the main text) are quite similar.

As with PCFG-based attacks, in our implementation we use a max-heap to store the interim results to maintain efficiency. To produce $k = 10^7$ guesses, we employ the strategy of first setting a lower bound (i.e., $10^{-9}$) for the probability of guesses generated, then sorting all the guesses and finally selecting the top $k$ ones. In this way, we manage to reduce the time overheads by 170% at the cost of about 110% increase in storage overheads, as compared to the strategy of producing exactly $k$ guesses. In Laplace Smoothing, it is required to add $\delta$ to the count of each substring and we set $\delta = 0.01$ as suggested by Ma et al. [8].

**Algorithm 1:** Markov-based password guess generation using laplace smoothing and end-symbol normalization

```
Input: A training set TS; The max password length maxLen; An
       estimation of lower bound of probability lowProb used in
       password guess generation; The markov-chain order mkOrder;
       A parameter k indicating the desired size of the guess list
Output: A password guess list L with the k highest ranked items
Training:
    for password ∈ TS do
        for i ← 1 to length(password) do
            preStr ← subStr(password, max(0, i − mkOrder), i − 1)
            nextChar ← getChar(password, i)
            trainingResult.insert(preStr, nextChar)
        preStr ← tailStr(mkOrder)
        nextChar ← end-symbol
        trainingResult.insert(preStr, nextChar)
Laplace smoothing:
    function trainingResult.getProbability(preStr, nextChar)
        count = trainingResult.getCount(preStr, nextChar) + 0.01
        countSum = trainingResult.getCount(preStr, charSet) +
        0.01 * trainingResult.charSet.size()
        return count/countSum
function ProduceGuess(password, probability)
    if probability ≥ lowProb then
        if getChar(password, probability) = end-symbol or
        length(password) ≤ maxLen then
            guessSet.insert(password, probability)
        else
            preStr ← tailStr(mkOrder)
            for char ∈ trainingResult.charSet(preStr) do
                newPassword ← password + char
                newProbability ← probability *
                trainingResult.getProbability(preStr, nextChar)
                if newProbability < lowProb then
                    continue
                ProduceGuess(newPassword, newProbability)
Produce k guesses:
    ProduceGuess(null, 1)
    L ← guessSet.top(k)
```

### 4.1 A subtlety about Good-Turing smoothing on password cracking

There is a subtlety to be noted when implementing the Good-Turing (GT) smoothing technique. We denote $f$ to be the frequency of an event, and $N_f$ to be the frequency of frequency $f$. According to the basic GT smoothing formula, the probability of a string "$c_1c_2\cdots c_l$" in a Markov model of order $n$ is denoted by

TABLE 3
Coverage of L (CoL) segments in corresponding test sets ("PY" stands for Pinyin and "1M" stands for one million)

| Test set | CoL (PY_name) | CoL (Duowan1M) | CoL(PY_name)∩ CoL(Duowan1M) | CoL(PY_name)− CoL(Duowan1M) | CoL(Duowan1M) −CoL(PY_name) | CoL (Duowan) | CoL(PY_name) ∩CoL(Duowan) | CoL(PY_name) −CoL(Duowan) | CoL(Duowan)− CoL(PY_name) |
|---|---|---|---|---|---|---|---|---|---|
| Tianya | 16.63% | 67.53% | 11.82% | **4.81%** | 55.71% | 74.34% | 13.75% | **2.88%** | 60.59% |
| 7k7k | 16.70% | 71.60% | 12.35% | **4.35%** | 59.25% | 79.84% | 14.49% | **2.20%** | 65.35% |
| Dodonew | 15.76% | 75.79% | 11.79% | **3.97%** | 63.99% | 81.19% | 13.47% | **2.29%** | 67.72% |
| 178 | 20.30% | 79.15% | 15.42% | **4.88%** | 63.73% | 83.98% | 17.49% | **2.81%** | 66.49% |
| CSDN | 17.26% | 65.64% | 11.35% | **5.90%** | 54.28% | 72.70% | 13.43% | **3.83%** | 59.27% |
| Duowan | 18.06% | 80.05% | 14.38% | **3.68%** | 65.67% | 100.00% | 18.06% | **0.00%** | 81.94% |
| Duowan_rest | 18.07% | 75.03% | 13.46% | **4.61%** | 61.57% | 100.00% | 18.07% | **0.00%** | 81.93% |

$$P(\text{“}c_1c_2\cdots c_{l-1}c_l\text{”}) = \prod_{i=1}^{l} P(\text{“}c_i|c_{i-n}c_{i-(n-1)}\cdots c_{i-1}\text{”}), \quad (1)$$

where the individual probabilities in the product are computed empirically by using the training sets. More specifically, each empirical probability is given by

$$P(\text{“}c_i|c_{i-n}\cdots c_{i-1}\text{”}) = \frac{S(count(c_{i-n}\cdots c_{i-1}c_i))}{\sum_{c\in\Sigma} S(count(c_{i-n}\cdots c_{i-1}c))}, \quad (2)$$

where the alphabet $\Sigma$ includes 10 printable numbers on the keyboard plus one special end-symbol (i.e., $c_E$) that denotes the end of a password, and $S(\cdot)$ is defined as:

$$S(f) = (f+1)\frac{N_{f+1}}{N_f}. \quad (3)$$

It can be confirmed that this kind of smoothing works well when $f$ is small, yet it fails for passwords with a high frequency because the estimates for $S(f)$ are not smooth. For instance, `12345` is the most common 5-character string in the Rockyou dataset and occurs $f = 490,044$ times. Since there is no 5-character string that occurs 490,045 times, $N_{490045}$ will be zero, implying the basic GT estimator will give a probability 0 for $P(\text{“}12345\text{”})$. A similar problem regarding the smoothing of frequency of passwords has been identified in [9].

There have been various improvements suggested in linguistics to cope this problem, among which is Gale and Hill's "simple Good-Turing smoothing" [10]. This improvement is famous for its simplicity and accuracy. This improvement (denoted by SGT) takes two steps of smoothing. Firstly, SGT performs a smoothing for $N_f$:

$$SN(f) = \begin{cases} N(1) & \text{if } f = 1 \\ \dfrac{2N(f)}{f^+ - f^-} & \text{if } 1 < f < \max(f) \\ \dfrac{2N(f)}{2f - f^-} & \text{if } f = \max(f) \end{cases} \quad (4)$$

where $f^+$ and $f^-$ stand for the next-largest and next-smallest values of $f$ for which $N_f > 0$. Then, SGT performs a linear regression for all values $SN_f$ and obtains a Zipf distribution: $Z(f) = C \cdot (f)^s$, where $C$ and $s$ are constants resulting from regression. Finally, SGT conducts a second smoothing by replacing the raw count $N_f$ from Eq.3 with $Z(f)$:

$$S(f) = \begin{cases} (f+1)\dfrac{N_{f+1}}{N_f} & \text{if } 0 \le f < f_0 \\ (f+1)\dfrac{Z(f+1)}{Z(f)} & \text{if } f_0 \le f \end{cases} \quad (5)$$

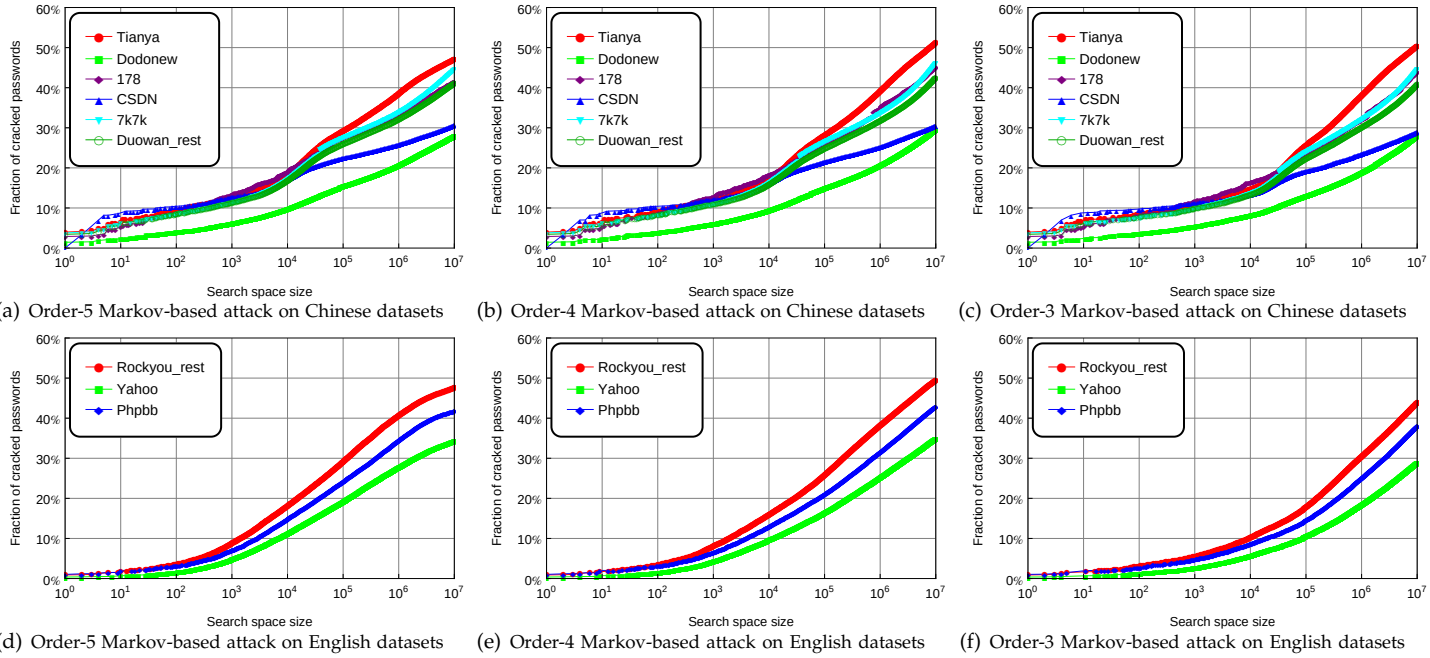
where $t(f) = |(f+1)\cdot\frac{N_{f+1}}{N_f} - (f+1)\cdot\frac{Z(f+1)}{Z(f)}|$ and $f_0 = \min\left\{f \in \mathbb{Z} \middle| N_f > 0, t(f) > 1.65\sqrt{(f+1)^2\frac{N_{f+1}}{N_f^2}(1+\frac{N_{f+1}}{N_f})}\right\}$.

In 2014, Ma et al. [8] introduced GT smoothing into Markov-based attacks to facilitate more accurate generation of password guesses, yet little attention has been paid to the unsoundness of GT for high frequency events as illustrated above. To the best of our knowledge, we for the first time well explicate the combination uses of GT and SGT in Markov-based password cracking.

## 5 Markov-Chain-based Cracking Results

The cracking results for attacking Scenario 1 have been included in Fig.5 of the main text, and the experiments for the three remaining scenarios are depicted in Fig.1, 3 and 2, respectively.

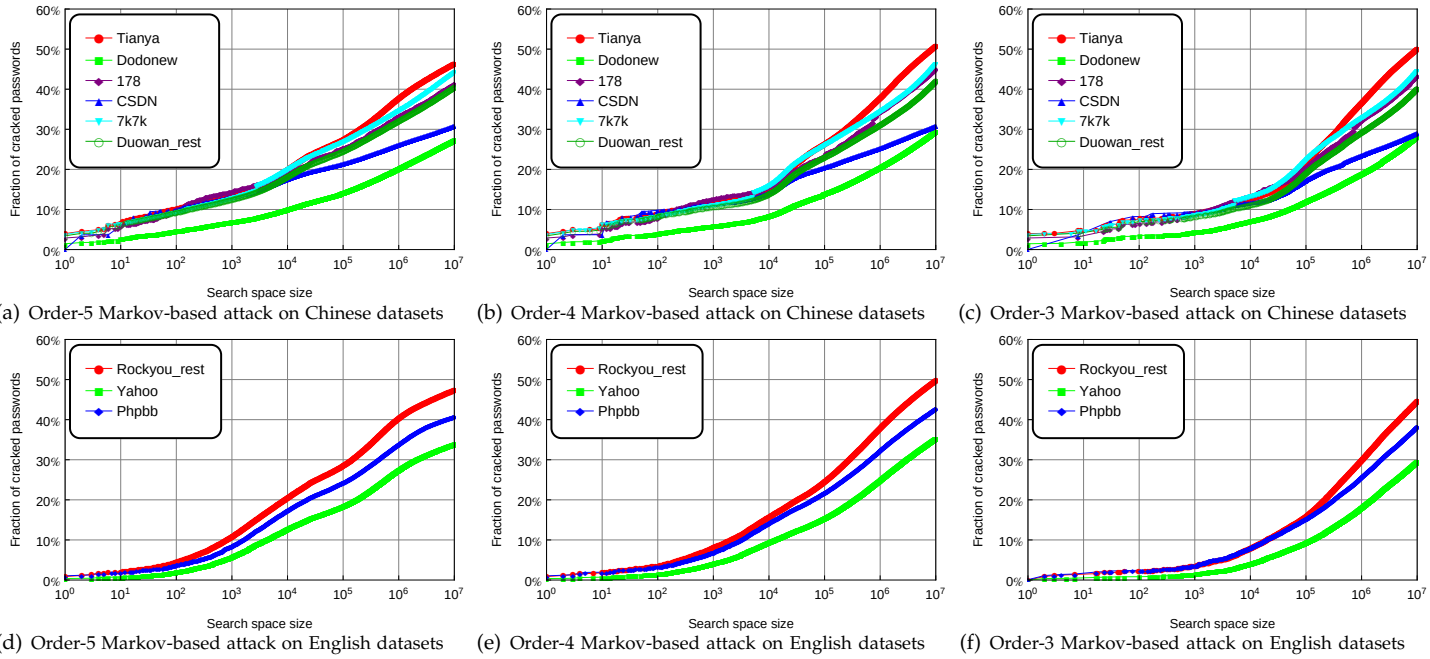
From Fig.5 of the main text and the figures 1, 2 and 3, one can see that for both Chinese and English test sets: (1) At large guesses (i.e., no less than $2 * 10^6$), order-4 markov-chain evidently performs better than the other two orders, while at small guesses (i.e., less than $10^6$) the larger the order, the better the performance will be; (2) There is no much difference in performance between Laplace Smoothing and Good-Turing Smoothing at small guesses, while the advantage of Laplace Smoothing gets greater as the guess number increases; (3) End-symbol-based normalization always performs better than the distribution-based approach, while at small guesses its advantages will be more obvious. This suggests that at large guesses, the attacks preferring order-4, Laplace Smoothing and end-symbol-based normalization perform the best among all the series of Markov-chain-based attacks, while at small guesses (e.g., less than $10^6$), the attacks preferring order-5, Laplace Smoothing and end-symbol-based normalization perform the best among all the series of Markov-chain-based attacks. It is worth noting that, the "reversal principle" also applies in all the markov-chain-based experiments. For example, in order-4 markov-chain-based experiments (see Fig.2(b) and Fig.2(e)), we can see that, when the guess number is below about 7000, Chinese web passwords are generally much *weaker* than their English counterparts. For example, at 1000 guesses, the success rate against Tianya, Dodonew and CSDN is 11.8%, 6.3% and 11.6%, respectively, while their English counterparts (i.e., Rockyou, Yahoo and Phpbb) is merely 8.1%, 4.3% and 7.1%, respectively. However, when the guess number is allowed to be over $10^4$, Chinese web passwords are generally *stronger* than their English counterparts. For example, at 1 million guesses, the success rate against Tianya, Dodonew and CSDN is 38.2%, 20.4% and 25.4%, respectively, while their English counterparts is 38.6%, 24.8% and 32.3%, respectively.




(a) Order-5 Markov-based attack on Chinese datasets (b) Order-4 Markov-based attack on Chinese datasets (c) Order-3 Markov-based attack on Chinese datasets

(d) Order-5 Markov-based attack on English datasets (e) Order-4 Markov-based attack on English datasets (f) Order-3 Markov-based attack on English datasets

Fig. 1. Markov-chain-based attacks on different groups of datasets (using *Laplace Smoothing* and *Distribution-based Normalization*). Attacks (a)∼(c) use 1M Duowan passwords as the training set, while attacks (d)∼(f) use 1M Rockyou passwords as the training set.




(a) Order-5 Markov-based attack on Chinese datasets (b) Order-4 Markov-based attack on Chinese datasets (c) Order-3 Markov-based attack on Chinese datasets

(d) Order-5 Markov-based attack on English datasets (e) Order-4 Markov-based attack on English datasets (f) Order-3 Markov-based attack on English datasets

Fig. 2. Markov-Chain-based attacks on different groups of datasets (using *Good-Turing Smoothing* and *End-Symbol Normalization*. Attacks (a)∼(c) use 1 million Duowan passwords as the training set, while attacks (d)∼(f) use 1 million Rockyou passwords as the training set.)

## References


[1] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *Proc. 30th IEEE Symp. on Security and Privacy*. IEEE, 2009, pp. 391–405.

[2] R. A. Butler, *List of the Most Common Names in the U.S.*, Jan. 2014, http://names.mongabay.com/most_common_surnames.htm.

[3] J. Goldman, *Chinese Hackers Publish 20 Million Hotel Reservations*, Dec. 2013, http://www.esecurityplanet.com/hackers/chinese-hackers-publish-20-million-hotel-reservations.html.

[4] *Sogou Internet thesaurus*, Sogou Labs, April 17 2014, http://www.sogou.com/labs/dl/w.html.

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

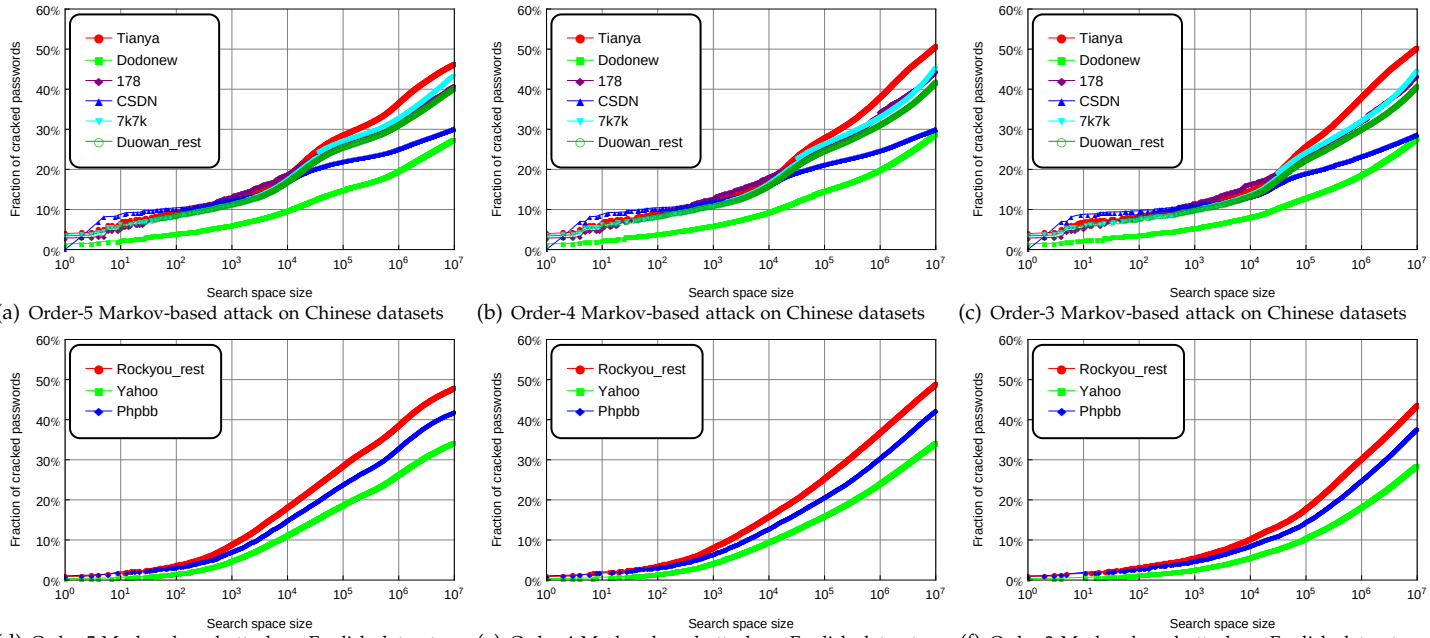
[6] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proc. CCS 2005*. ACM, pp. 364–372.

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

[8] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in *Proc. IEEE S&P 2014*. IEEE, 2014, pp. 538–552.

[9] J. Bonneau, "Guessing human-chosen secrets," Ph.D. dissertation, University of Cambridge, 2012.

[10] W. Gale and G. Sampson, "Good-turing smoothing without tears," *Journal of Quantitative Linguistics*, vol. 2, no. 3, pp. 217–237, 1995.




(a) Order-5 Markov-based attack on Chinese datasets (b) Order-4 Markov-based attack on Chinese datasets (c) Order-3 Markov-based attack on Chinese datasets

(d) Order-5 Markov-based attack on English datasets (e) Order-4 Markov-based attack on English datasets (f) Order-3 Markov-based attack on English datasets

Fig. 3. Markov-Chain-based attacks on different groups of datasets (using *Good-Turing Smoothing* and *Distribution-based Normalization*). Attacks (a)∼(c) use 1 million Duowan passwords as the training set, while attacks (d)∼(f) use 1 million Rockyou as the training set.