Leet Usage and Its Effect on Password Security

Wanda Li and Jianping Zeng

***Abstract*—Text-based passwords have long acted as the dominating authentication method. Leet, as one of the significant components in password, has not been paid enough attention yet. In this paper, we systematically study the presence of Leet in passwords. We define single and pattern forms of Leet and propose a matching approach to check whether a user password contains Leet. We extract the most prevalent counterpart pairs of Leet manifestations. Afterward, we examine the effect of Leet in passwords by incorporating Leet transformation into the probabilistic context-free grammar(PCFG) method to crack passwords. We construct the first comprehensively analyzed dictionary of Leets for passwords, which is confirmed suitable for most datasets by user survey. Experiments on four leaked password sets demonstrate that distinguished Leet usage accumulates to account for around 1% of the total dataset. Only 5% of high-frequency Leets replacement could increase the cracking rate by 0.55%. For crackers, incorporating popular Leets aids to improve password cracking performance. For users, adopting low-frequency Leets could strengthen their passwords. This research provides a new perspective to investigate Leet transformations in passwords.**

***Index Terms*—Password analysis, password attack and protection, pattern matching.**

I. Introduction

TEXT passwords are especially ubiquitous in authentication systems and act as the foundation of security policy for a broad spectrum of online services, such as personal financial transactions and communications protection. Unfortunately, human users are inclined to choose weak passwords because it is easier to remember them [7], [20]. Such limited human memorability leads to poor randomness of passwords, which makes them the weakest link in the authentication chain [10], [35]. What makes things worse is the increasing number of passwords a user has to manage [19]. Therefore, users often cope by reusing passwords across accounts on different online services. Consequently, the majority of passwords crowd in a small portion of password space, which empowers brute-force or dictionary attacks.

Manuscript received December 5, 2019; revised March 13, 2020 and July 8, 2020; accepted December 22, 2020. Date of publication January 8, 2021; date of current version February 1, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0803203 and Grant 2016YFB0800101, in part by the Shanghai Municipal Natural Science Foundation under Grant 15ZR1403700, in part by the CURE (Hui-Chun Chin and Tsung-Dao Lee Chinese Undergraduate Research Endowment) under Grant 19015, and in part by the FPE of Science and Technology Development Center, Ministry of Education, China, under Grant 2017A03021. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Karen Renaud. *(Corresponding author: Jianping Zeng.)*

The authors are with the School of Computer Science, Fudan University, Shanghai 200433, China, and also with the Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, Shanghai 200433, China (e-mail: wdli16@fudan.edu.cn; zjp@fudan.edu.cn).

Digital Object Identifier 10.1109/TIFS.2021.3050066

Most online services nowadays provide password meters or employ stricter password composition policies to help users to know the strength of their passwords. Researches have shown that password meters and composition policies could work together to help users to choose stronger passwords [29], [45], [50]. Accurately assessing passwords strength requires a deep understanding of the exact tactics for users to construct their passwords. Also, such principles help crackers perform more efficiently. At the same time, users could intently avoid using weak methodologies to create passwords with the awareness of commonly used methodologies.

Critical efforts have been made to unveil the structure of passwords. The most initial dictionary attacks have proved that users are in favor of simple dictionary words [39], or at least phonetically memorable sequences [40] when constructing passwords. Weir *et al.* [57] developed a template-based password model that uses PCFG to describe password structure. As studies in the underlying distribution of user-generated passwords have been launched [54], we can see that some elements in daily life are pervasive, like keyboard sequence such as "qwert" or "qawsed" [26], personal information like birthdate and name, e.g., "19801215" and "Mary" [30] and trivial sequences as "password" and "123456" [31].

This paper studies another common component in passwords, i.e., the Leetspeak (Leet, or "1337"). According to the investigation of state-of-the-art literature, we find that existing password cracking methods have not yet put enough attention to Leetspeak usage in passwords. Although some works mention Leets [1], [6], [17], [47], Leet have not been analyzed concentratedly, and only a few of Leets are considered in most cases. Our goal is to find out the most likely and popular Leet patterns and characters in passwords, and reveal their effect on password strength, i.e. whether to incorporate Leet with existing attacking methods could improve cracking performance. In this work, we choose a prevalent cracking framework, PCFG [26], [30], [36], [52], [9], [32], as an example of base cracking methods. This is because rule-based systems are able to match or outperform other password guessing tools when the number of allowed guesses was small [25]. Although deep learning-based methods are thriving as an alternative recently, we have to point out that their explainability is too blurred for this work.

The most significant hindrance for us to identify a Leetspeak is its intricacy nature. The particular grammars, terminologies and even spellings make Leet rather like a hacker slang. To tackle this, we utilize a thorough Leet dictionary in cracking process of several leaked password datasets from websites of different languages. First, we develop a method to gauge possible Leetspeak usage in password creation. After analyzing their frequency, we obtain the most popular Leets used in passwords, both in single and pattern form. It comes out that many little-known Leets are still commonly used.

Our contributions can be summarized as follows:

- We believe this is the first systematic study of Leetspeak replacements in passwords with single and pattern forms, introducing the Leets in passwords by the definition, detection, dictionary construction, and usage. This work on Leetspeak transforming quantification and password cracking could be applied to any other text-based password datasets from different websites.
- We propose a method to discover both forms of Leet usage in passwords. The method considers the length of patterns by choosing a proper threshold in *n* of n-grams, which improves the performance.
- We show that Leet method could work as a double-edged sword in passwords. On the one hand, common Leet transformations could be easily incorporated into existing methods like PCFG for password cracking, and enhance their performance in the long run. On the other hand, low-frequency Leets could help users to strengthen their passwords.

The remaining sections are organized as follows. We describe the related researches in the next section. In Section III, we cover details of the approach for analyzing Leet usage in passwords. We describe the password sets, present the experiment results, and discuss the analysis method in Section IV. Section V introduces our survey and the summation of its results. Then we discuss some implications and limitations of our research in Section VI, and finally, draw conclusions and point out future work in Section VII.

II. Related Work

A. Password Study and Role of Leets

Through decades, authentication using text-based passwords remains the de facto standard for authentication in today‘s Internet. However, early researchers like Morris and Thompson [39] pointed out that passwords are so simple that always vulnerable to guessing attacks. As a result, a substantial number of previous works have been done to understand user-created passwords, measure their strength and enhance their security. Passwords have many inspiring characteristics. For instance, Li *et al.* [31] studied more than 100 million real-life Chinese passwords and presented differences between passwords in Chinese and other languages. Kiesel *et al.* [28] analyzed several corpora to reveal that distributions of mnemonic passwords can reach the same strength against offline attacks with fewer characters. Analysis considering human intrinsic are also launched, such as habits [24], general inspirations [49], semantic patterns [51], and security awareness [48].

Another active research area in recent years is to study the quality of user’s passwords. As Shannon’s entropy can somewhat measure random level in cryptology accurately [14], [43], [10], [27], many other metrics have been introduced, such as marginal guesswork [43] and marginal success rate [12]. Empirical studies (e.g., [27], [59]) got conveyed to conquer the measuring problem. Knowing the strength of passwords is useful to improve password strength meters, an important tool to help users choose secure passwords. Studies like [15], [18], [22] analyze a bundle of meters and offer profitable insights.

All efforts above base on characters or patterns in passwords. Leet, as one of the most common phenomena in passwords, to which study could shed light on studies of password analysis, cracking, and strength measurement. Unfortunately, existing password cracking methods have not yet put enough attention to Leetspeak usage in passwords. Although [47] investigated its use via crowd-sourcing, they did not mention details about Leets, e.g. how many Leets were involved, how to discover them, and to what extent they could influence the strength of passwords. Leets have been only acting a minor part, hiding in the subsisting list. In this work, we want to analyze Leet usage in passwords comprehensively, and reveal the scope and extent of its effect.

B. Password Cracking

The discussion of password-cracking strategies comes almost at the same time as the password itself. For an attacker, brute force and dictionary attacks are the most basic choices. If typical users’ habits in the choice of passwords are taken into account, dictionary attacks will be found effective [39]. As an effort to diminish redundant work, Oechslin proposed Rainbow table [41], which reduces by two the number of calculations needed during cryptanalysis.

Nevertheless, password policy now becomes stricter, promoting the creation of more powerful attacking means. Password cracking approaches that use dictionaries (i.e., John the Ripper(JtR) [6] and HashCat [1]), generally use them to create guesses by using the dictionary entries and then mangling these entries in some systematic ways. Such methods generate a limited number of guesses and require relatively more time. Other than that, various technologies based on the probabilistic model have been applied. Narayanan and Shmatikov [40] employed Markov chain to create a probabilistic approach. Castelluccia *et al.* [16] improved cracking efficiency by proposing OMEN based on the model in [40]. PCFG proposed by Weir *et al.* [57] trains part of a revealed password sets to generate grammars and later, in turn, generates guesses by this grammar. As probability-threshold graphs are better tools than guess number graphs [34], it has been acted as the basis of studies like [9] and enriched from different perspectives [26], [30], [36], [52].

Most recently, researchers like Melicher *et al.* [37] and Hitaj *et al.* [25] introduced neural networks to guess passwords. Liu et al [32] proposed GENPass, which integrates LSTM with PCFG, to improve matching rate in both one-site and cross-site tests. Different from traditional ideas, neural networks hinder the details of generation but output an infinite number of guesses. Although they could outperform traditional methods, they have to guess more times, which hinders their practicality [25].

C. Patterns in Passwords

PCFG defines the most essential password pattern(**L**, **S**, and **D**) to form password templates, and demonstrates how to "learn" password patterns from known password distributions. Users are long known to use specific sorts of information in passwords. Bonneau *et al.* [11] found from banking customers that sharing and reusing PINs are primarily based on the victims' birthday. Schweitzer *et al.* [44] investigated how keyboard patterns are used in passwords. Han *et al.* [23] studied the differences between passwords from Chinese and English-dominant users, mainly considered regional patterns. Thus, further detailed categories that can better portray how people choose their passwords are added in PCFG to improve guessing performance. For instance, Houshmand *et al.* [26] added keyboard patterns and multiword patterns (two or more words in the alphabetic part of a password) systematically to the context-free grammars used in the probabilistic password cracking. Li *et al.* [30] extended the PCFGs method to be semantics-rich from the perspective of personal information, counting personalized characteristics like name, Email address, and birth date as a new structure in passwords. Also, researchers like Zeng *et al.* [60] conducted research on large-scale password sets and found that lexical sentiment, primarily, positive and joy type can be utilized as a component in password patterns. Nevertheless, although transformations like "password" to "p@ssword" has always been a common example in the field, only a few pieces of research have put a real notice to people's usage of Leetspeak. How Leet is practiced in password remains a real question that is worth quantifying, for they still consist of an inevitable part of password transformation [17]. No empirical study on specific transformation to the existing passwords has been performed yet, which motivates us to dig deeper.

III. Approach

In this section, we introduce our main process of attacking with Leet. As a preparation, we create a splitting dictionary to distinguish Leets in Section III-A, then define Leets in passwords in Section III-B.

Our approach has three sessions, as shown in Figure 1. Here we randomly select a half of passwords in each dataset as the training set to detect Leets and construct PCFG model. In Leet matching session (Section III-C), we pre-process the training set and detect Leets in it. This session outputs a dictionary of the most likely Leets, *Top Leets*. In the guessing session, we let PCFG guess on the training set, then leverage *Top Leets* to generate the final guessing results. In the cracking session, we perform attacks on the testing set. The last two sessions are explained in Section III-D. We describe how to match Leet and incorporate it into PCFG in detail as follows.

A. Creation of Splitting Dictionary

In order to distinguish possible Leet changes, one password string should firstly be split into meaningful units and other remaining segments. Note that passwords generally depend on individuals' choices, which can be primarily affected by people's native language and living environment; common

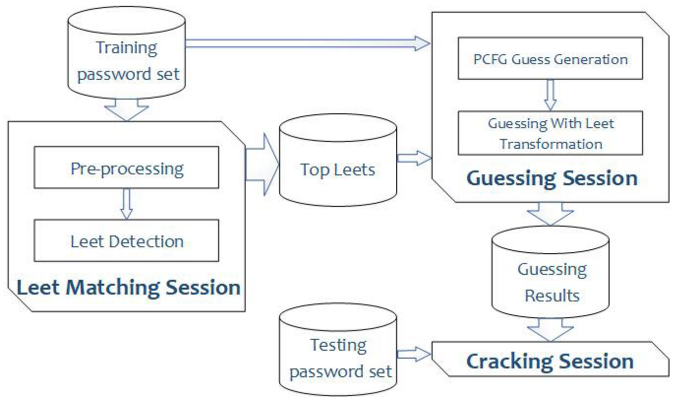


Fig. 1. Diagram of the process followed in Leet approach of the guessing and cracking phase.

dictionaries may not be comprehensive enough to cover all information. In consideration of this, we create a full dictionary **NormDic** for the splitting.

In this paper, we use large-scale leaked password sets from both English and Chinese websites. Thus, in addition to ordinary English words, Chinese Pinyins and other spellings like Chinese family names are also included in the dictionary.

The first step of creating **NormDic** is to combine four dictionaries *D1*, *D2*, *D3* and *D4*. We describe all of them below inside out.

1) D1: a list of 15,000 English words which are top frequently used in British National Corpus (BNC [5]). BNC is a 100-million-word text corpus of samples of written and spoken English from a wide range of sources. We selected the words in *D1* based on their occurrences in BNC, so different forms of words, including single, plural, tenses, might appear in the dictionaries. For example, "unequal", "equally", "equality", "unequally" along with "equal" are all included.

2) D2: a list of 399 spellings for 4761 frequently used Chinese words. Note that many Chinese words share the same spellings. For instance, "shang" has several meanings, including "up", "quotient", "entropy", and so on. Hence, we do not need too much spelling to represent a rich set of common Chinese words.

3) D3: a list of top 209 frequently used Chinese family names. Name information is a common component of passwords, both in the English context [58] and Chinese context [30].

4) D4: a list of manually selected entities, such as abbreviations for famous corporation names, popular Internet slogan. Some of them are "avon", "intel", "sony", and "volvo". We add *D4* to **NormDic** so as to increase its comprehensiveness.

Then, we combine all of the items in *D1*, *D2*, *D3*, *D4* and a frequently used password dataset into the new dictionary **NormDic**, which turn out to have more than 11,000 words.

B. Define Leets in Password

Leet (or "1337"), also known as eLeet or Leetspeak, is a system of modified spellings used primarily on the Internet [2]. It has a system of suffixes and alternate meanings, which is usually existed as the basis of word modifying. Generally,

TABLE I
SAMPLE OF LEETDIC

Original Word(w_o)	Leet
A	4, @, aye
D	|), cl
O	0, (), <>
S	$, z, 5
W	\/\/, vv, 2u

it uses character replacements in ways that play on the similarity of their glyphs via reflection or other resemblance. Different online communities together have created many dialects or linguistic varieties. In this paper, we adopt the most exhaustive Leet list spreading on the Internet [3] as **LeetDic**. It includes 227 possible representations, some of them are shown in Table I. Note that both sides of representation might have more than one meaning. For example, "A" can be written as "4" and "@", while "S" can be represented by "$" or "5". Also, "|" could be interpreted by both "I" and "l".

To simplify, a Leet's corresponding letter in the normal exhibition is referred to as the original word (w_o), and p_o denotes the original password, which is the password before the user conducts Leet transformation in it. To put Leet transformations into effect, we add an additional Leet-like phrase matching phase and an adaptive-substitution phase after the primary PCFG method.

In this paper, we novelly propose the notion of the Leet pattern, which is a sequence of characters that contains Leet letters. Though seemingly confused at first glance, they construct an innovative pattern that has the same meaning as the word it initially is.

A typical example is **pa$$w0rd**, which will be abbreviated to a complicated base structure $L_2S_2L_1D_1L_2$ in PCFG. However, in Leet pattern, **pa$$w0rd** will be spotted as a whole pattern. Such patterns are often combined with other components to create the password, for example, **pa$$w0rd1517**. Because both "$" and "0" are Leet words, and "password" is in **NormDic**, **pa$$w0rd** can be identified. Table II shows some examples of Leet patterns in passwords.

Specifically, we try to find exact occurrences of the Leet patterns in the passwords. A match is considered found if and only if the following conditions are satisfied: a sequence including Leet(s) is in the password, and after replacing the Leet(s) with proper normal letters, the sequence can be spotted in **NormDic**. In password study, some researchers choose not to match dictionary words whose length is less than 4 to ensure accuracy [34]. Nevertheless, this method is likely to result in underfitting, especially in our context. Barely ignore short grams is not a proper choice for the following reasons. First, Chinese Pinyins, as well as English words, have a large quantity of 2-grams and 3-grams. Unfortunately, people are known to use words from their first language in their passwords [10]. Second, 3-grams account for a particular part in our dictionary **NormDic** (3-grams occupies 4.84%, and the remaining length, 93.97%). Third, studies also show that people tend to use the shorter or abbreviated word in passwords [8], [38]. In order to test the influence of including 3-grams while detecting Leet patterns, we later compare the number of Leets we could detect and the accuracy of detection under both conditions.

TABLE II
SOME SAMPLE PASSWORDS CONTAIN LEET AND PATTERN IN THEM

Original Password(p_o)	Password	Normal word	Leet pattern
loveyou	lov3you	love	lov3
softball	softba11	ball	ba11
greenday	greend@y	day	d@y
nopass	nopa55	pass	pa55
anaconda$	anac0nda$	anaconda	anac0nda

TABLE III
INSTANCES OF PCFG PASSWORD STRUCTURES

Structure	Example
Simple	LDS
Base	$L_8D_3S_1$
Pre-terminal	$L_8123@$
Terminal	$dearbook123@$

C. Detecting and Matching Leet

In this section, we describe our methodologies to deal with Leets in the training data, including data cleaning and the distinguishing of Leets. To properly measure the degree of regular dictionary words' involvement in an individual password, we introduce Coverage, a metric to quantify the extent of **NormDic** word occupies in a given password. We also propose a matching method to find possible Leet patterns and single Leet replacements in a user password.

The task of matching Leet patterns could be addressed by using a dictionary. However, to the best of our knowledge, no existing dictionary has differentiated single Leetspeak characters by their probability of occurrence, nor did Leet patterns. As an alternative, we chose to solve the problem through modifications of the probabilistic context-free guesses. In other words, extra Leet matching and replacing are integrated into the initial training and cracking phase additionally.

1) Pre-Processing: Due to the tangled nature of passwords, we conduct a pre-process before we start to match Leets in each of the leaked password datasets. We first employ the concepts presented in [57] to define the plain structure. Three kinds of symbols are used to express the basic structure: an *alpha string*(**L**) be a sequence of alphabet symbols, a *digit string*(**D**) be a sequence of digits, and a *special string*(**S**) be a sequence of non-alpha and non-digit symbols. Table III shows different structures of PCFG.

In this paper, we define the plain structure as an all-digital structure. In other words, a plain password is composed of all digits, and in a normal length of a password (in this paper, it should be longer than 3 and less than 40 in length). For instance, **6569123** is a plain structure password, but **1password23**, which can be denoted as $D_1L_8D_2$, is not a plain structure one. Passwords with the plain structure are excluded in this work for efficiency issues. We also disregard those without character in **LeetDic** because they do not fulfill the precondition of Leet transformation.

2) Coverage: By comparing the Coverage value of the user password containing Leet and its corresponding p_o, we could determine whether a password replacement is a valid Leet transformation. We use this measure in a similar way to [30], where it is used to quantify the correlation between personal information and user passwords.

Computing Coverage requires splitting password strings, which is a challenging task. The difficulty lies in that the splitting might have different results, and it is not easy to determine which one is the best p_o. Our strategy is to match words in **NormDic** as long as possible in order to avoid triviality, because we want to find all of the possible longest match words in the given password.

Mathematically, we denote l_i as the total length of the matched word i after splitting, l_P as the length of the current sequence, and n is the number of matching words. Then let the Coverage rate C be

$$C = \sum_{i=1}^{n} \left(\frac{l_i^2}{l_P^2} \right) \tag{1}$$

The value of Coverage ranges from 0 to 1. The larger the Coverage is, the stronger a password is correlating with the dictionary. Coverage "0" means no dictionary word is detected in a password, and Coverage "1" indicates the entire password is perfectly matched with one word in **NormDic**. Algorithm 1 shows how to compute Coverage and find the maximum cover list. We take the password, *pwd*, and the shortest covering length, *covLen*, as input and make use of a dynamic programming strategy.

The algorithm could be divided into three steps. First (i.e. line 5 to line 11), we maintain a dynamic-sized window to find all possible matches in *pwd*. The initial length of the window is *covLen*, which will be 3 when we include 3-grams but 4 when not. The result of this step can be different based on *covLen*. All matched word's start and terminal index is recorded as a pair in *curList*, i.e. $curList[k][1]$ is the terminal index of the k-th pair in *curList*.

If there exist matches, we conduct the second step (line 16 to line 19). In this step, we adopt a dynamic programming approach to find the maximum cover segment set of the password. It ends up with start and terminal indexes pair(s) stored in *coverIndex*. Finally, we compute the Coverage of the input password by Equation 1 at line 22.

To better illustrate how the entire process is conducted, we use password "bluesky0" as an instance. It has a *curList* of [(0, 3), (0, 4), (4, 6)], which indicates that there are three possible words ("blue", "blues", "sky") in it if we consider 3-gram. However, when we do not include 3-gram (*covLen* is 4), *curList* will be [(0, 3), (0, 4)]. In this case, if the *covLen* equals to 3, *DP* will be [4, 4, 4, 4, 7, 7, 7] and we will get an *coverIndex* of [(0, 3), (4, 6)].

3) Leet Detection: To a certain password, pwd, we first find all the possible Leet letters in it referring to **LeetDic**. Then

Algorithm 1 findMaxCover($pwd, covLen$)

```
Input:
    The given password, $pwd$.
    The shortest covering length of words, $covLen$.
Output:
    The value of Coverage, $Coverage$;
    The array of maximum cover segment set index of $pwd$, $coverIndex$.
1: $curList \leftarrow NULL$
2: $coverIndex \leftarrow NULL$
3: $Coverage \leftarrow 0$
4: $pwdLen \leftarrow len(pwd)$
5: for $m = 0$ to $pwdLen - covLen$ do
6:   for $n = m + covLen$ to $pwdLen$ do
7:     if $pwd[m:n]$ in $NormDic$ then
8:       $curList.append([m, n])$
9:     end if
10:  end for
11: end for
12: if $curList$ is not $NULL$ then
13:   $curLen \leftarrow len(curList)$
14:   $DP \leftarrow [0] * curLen$
15:   $curDiff \leftarrow$ distance within each pair in $curList$
16:   for $k = 1$ to $curLen$ do
17:     $t \leftarrow curList[k-1][1] + DP[curDiff[k]]$
18:     $DP[k] \leftarrow max(t, DP[k-1])$
19:   end for
20:   $coverIndex \leftarrow curList$ pairs in $DP$
21:   for all $l_i$ such that $l_i \in coverIndex$ do
22:     $Coverage \leftarrow Coverage + len(l_i)^2/pwdLen^2$
23:   end for
24: end if
25: return $Coverage, coverIndex$
```

we try all permutations when replacing those detected Leet characters with their w_o (some characters can represent several letters) in **LeetDic**, and take one replacement into account if it increases the Coverage of pwd. For example, password "13luesky0" has a coverage rate computed using Equation 1 as

$$C = \sum_{i=1}^{1} \left(\frac{l_i^2}{l_P^2} \right) = \frac{3^2}{9^2} = 0.11.$$

whereas its valid p_o "bluesky0" has a coverage rate of

$$C = \sum_{i=1}^{2} \left(\frac{l_i^2}{l_P^2} \right) = \frac{4^2 + 3^2}{8^2} = 0.339.$$

In this case, we draw the conclusion that "bluesky0" is a p_o of "13bluesky0".

We use Algorithm 1 to analyze all of the passwords in the training set to detect all Leets getting used. Those who ranked top 20% among all Leets in one dataset are considered as the mostly used Leets which we call *Top Leets*. They are used in the cracking session as input. The frequency of Leet patterns follows a long-tailed distribution. A manually random examination shows that patterns fall into the tail are less likely to make sense. However, a rarely appeared pattern still costs as equal to a high-frequency one in the matching process. To balance accuracy and efficiency, we set a threshold of frequency to determine whether to take one pattern into the pattern dictionary. The threshold should lie in where the tail of distribution starts roughly, which in this case is 20. The results of the cracking experiment and user survey shown in Section V confirm this choice.

D. Leet Attacks

We use PCFG cracker [4] as the base guess generator to demonstrate the usage scenario of Leet transformation in the sequence of attacking. According to the extensive experiments conducted in [33], PCFG is among the most effective programs in terms of the speed of generating correct guess, which makes it suitable as a base method for illustrating Leet attacks. On the other hand, although methods like neural networks could automatically guess from scratch, their learned structures are not interpretable, which will not benefit the distinguishing of Leets. In addition to the formal guessing phase, we perform a Leet matching phase in order to count the frequency of Leet characters in the current training dataset.

1) PCFG Guess Generation: Weir *et al.* derive a probabilistic context-free grammar from training set in PCFG method. Based on various possibilities acquired at the training data, the grammar later helps to generate guesses in the order of frequency. When guessing, PCFG first produces the pre-terminal structure by filling specific values in the **D** and **S** parts of the base structure. The values are sorted in descending order of frequency in advance. One base structure may have multiple substitutions. Then, **L** part is fulfilled by the frequency of attached dictionaries since the space of alpha strings is too large to learn from the training set. As PCFG can produce statistically high-probability passwords first, it only needs to guess significantly fewer times than traditional dictionary attacks. Those guesses are hashed to compare with values in password databases. They act as the baseline to determine whether a matching result is eligible or not in the following steps.

2) Guessing With Leet Transformation: We feed the output of PCFG cracker and *Top Leets* to this phase as input. Assume we will attack T times, and the replacement rate isβ. Namely, we choose the top β percent of guessing passwords, which are more likely to appear in the real password set, as the raw material to conduct Leet transforming. Then we use the passwords after Leet replacement to substitute the same number of guessing results with the overall lowest possibility in PCFG, which is the last part of the T passwords.

We show how to generate guesses based on particular password in Algorithm 2. The high-level idea is that we detect possible Leet appearance in the password, then alter them to generalize guesses.

Algorithm 2 uses $TopLeet$, $covLen$ and a given password pwd as input. pwd can be a guess result of other tools. To start with, we examine whether Leet occupies a reasonable part of the input password. If the ratio of total Leet length to password length reaches drop-out ratio α, we skip this password. This is to the consideration of accuracy, for the permutation of different replacements of Leets in a sequence of pure Leets can generate completely unexpected sequences. To illustrate, after a sequence of transforming, the password "11132m" will have "literm" as a p_o. However, such an explanation does not seem to make sense for Leet characters occupied 83.33% of its total length. In this study, α is set to 0.7 as an empirical constant.

Then we try to create all combinations of distinct Leet change by arranging every original word w_o of Leet and the remaining parts of this password (line 9 to line 16). We use IPL to track the lists of possible w_os. In $sepByLeet$, we cut Leets and normal characters in pwd into a list of segments, sep. $LeetIndex$ stores the indexes of Leet segments in sep. Each index in $LeetIndex$ corresponds to a version of IPL, i.e. IPL_i. We use function $findMaxCover$ to get the splitting result of pwd, and store it in $maxCov$ and $maxPer$ respectively, which represent the baseline cover index list $CoverIndex$ and Coverage value. After that, we split each p_o and select the result which produces the maximum Coverage while has the relatively least splitting words. If there are more than one available result, we count in all of them. A password with larger Coverage is usually more vulnerable to

Algorithm 2 Generate Possible Original passwords(p_o)

```
Input:
    The set of Top Leets, $TopLeets$;
    The shortest covering length of words, $covLen$.
    The guess result, $pwd$.
Output:
    A set of possible original passwords of $pwd$, $POS$.
1: $Leets \leftarrow findLeet(pwd)$
2: $POS \leftarrow NULL$
3: if $len(Leets)/len(pwd) > \alpha$ or $Leets$ is $NULL$ then
4:   return $POS$
5: end if
6: $i \leftarrow 1$
7: $IPL_0 \leftarrow [\ pwd\ ]$
8: $sep, LeetIndex \leftarrow sepByLeet(pwd, TopLeets)$
9: for all $lt$ such that $lt \in LeetIndex$ do
10:   $IPL_i \leftarrow IPL_{i-1}$
11:   for all $p$ such that $p \in IPL_{i-1}$ do
12:     $q \leftarrow$ replace $sep[lt]$ with its $w_o$ in $p$
13:     $IPL_i.Append(q)$
14:   end for
15:   $i \leftarrow i + 1$
16: end for
17: $maxCov, maxPer \leftarrow findMaxCover(pwd, covLen)$
18: for all $p_o$ such that $p_o \in IPL_i$ do
19:   $Cov_k, Per_k \leftarrow findMaxCover(p_o, covLen)$
20:   if $Per_k \geq maxPer$ and $Cov_k \neq maxCov$ then
21:     $POS.Add(p_o)$
22:   end if
23: end for
24: return $POS$
```

TABLE IV
THE PROPORTION OF PASSWORDS WHICH ONLY CONTAIN NUMBERS AND THE PROPORTION OF UPPERCASE LETTERS IN ALL CHARACTERS IN THE DATASETS

Dataset	Proportion of pure number Passwords	Upper Case Letter Percentage
Gmail	15.289%	0.000%
RockYou	16.359%	4.354%
CSDN	44.184%	1.952%
T178	48.074%	0.457%

dictionary attacks, since it is prone to have a high-frequency PCFG password structure. Considering users' motivation of using Leet, a p_o will only be accepted as a valid original password when it has a Coverage at least equal to the user password. At the same time, their cover index list should be unequal. Every p_o meets this requirement will be added to POS.

IV. EXPERIMENT AND RESULTS

A. Datasets

Several previous password studies used leaked password sets in the experiments since the number of passwords can be large enough for statistical analysis. Similarly, we employ four large-scale password sets that were leaked in recent years. All of the data sources we used were publicly announced at least once, and none had the veracity questioned by the affected website. We only acquire the passwords without their corresponding account information. Verifying the validity of the data directly by attempting to use the credentials would be unethical. Those password sets are described as follows.

CSDN(Chinese Software Developer Network) is a large online forum like GeeksforGeeks in China. Its users are mainly software developers, electronics engineers, and computer science students. This password dataset leaked on an attack to csdn.net in 2011. Gmail is a widely used email site, which has a distributed source of users. RockYou is an online social games company that developed widgets for MySpace. It was cracked in 2009. T178 is an online game website that provides several kinds of games, which has users such as office staff and housemakers. Hence, by selecting leaked passwords from websites of different functions and user communities, the sets can be representative enough for our investigation on the usage of Leets in passwords. Table IV shows the features about the proportion of passwords that only contain numbers and the percentage of uppercase letters in all characters from the password sets. Note that although the Gmail dataset has no upper case letter, it does not blur the results as we discussed above. All of the four sets were randomly separated into two even parts. One is used as training set, and the other is testing set. The training and test set are mutually exclusive. Note that the training set is exactly the set used in the former section.

- CSDN: 4,411,114 passwords, including 2,206,893 as training set.
- Gmail: 4,929,068 passwords, including 2,456,952 as training set.
- RockYou: 14,344,392 passwords, including 7,174,651 as training set.
- T178: 9,072,966 passwords, including 4,537,419 as training set.

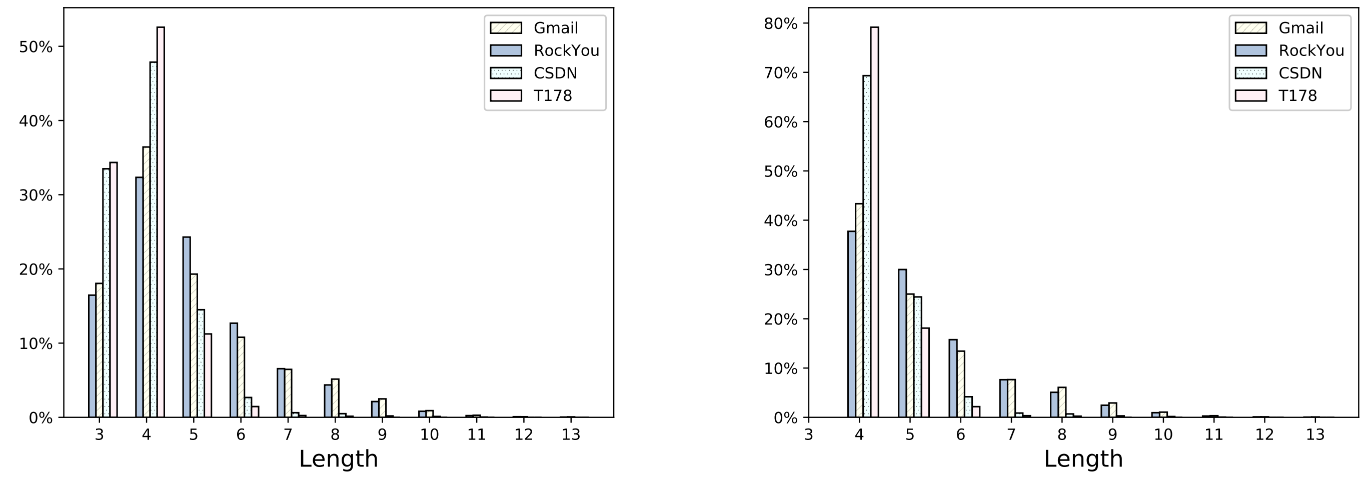
Upper case characters only occupy a minor fraction of all of our datasets, and we have reasonably inferred this as a common phenomenon. Because of this, we lowercased all input words in the training set before using them. Correspondingly, all letters in **LeetDic** and **NormDic** are in lower case, too. Note that PCFG does not distinguish upper case letters and lower case ones likewise. Thus, lowering case to all training set not only has a slight effect on the result but enhance the efficiency of detecting Leets by deducing the comparison times.

B. Empirical Study on Leet Usage in Passwords

After splitting passwords, replacing Leets in possible patterns, and linking to our dictionary **NormDic**, we can get two lists. One is of Leet patterns, denoted as $L_p = (pair_i, pat_i)$, where $i = 1, 2, \ldots$, M, M is the number of characters happened in matching Leet patterns. Here $pair_i$ is the ith pair of Leet and its w_o, pat_i is a list of patterns sorted by frequency of appearance. The other is of Leet characters being used in the current set, denoted as $L_C = (pair_i, occ_i)$, in which occ_i means the total possible occurrence times of the ith pair. Note that occ_i is an estimated static to some extent because a Leet can represent several different normal letters, while a normal letter could be replaced by more than ten Leets sometimes. We perform statistical analysis with and without including 3-grams on all of our datasets, then describe the results as follows.

In Figure 3 we present different dataset's proportion of used Leet in **LeetDic**. We can see that although there are 227 characters in **LeetDic**, frequently used ones only account for about 21% to 35%, depending on the dataset and the number of grams we considered. Different portions of Leet pattern with distinctive length represent in all Leet patterns are shown in Figure 2. The percentage of usage times for Leet is defined as the ratio of appearances of words in L_P to the total number of occurrences of English words and Chinese Pinyins in the respective training set, which ranges from 1.2% (Gmail) to 0.6% (T178). If a password segment matches a word in the dictionary after Leet character in it is replaced with the corresponding letter(s), we count it as a variant usage. Note that one Leet can have more than one candidate leading to match. On such a condition, we regard it as appearing once instead of multiple times. In the meantime, being inspired by [57], for the exact Leet character in one password segment, each matching word in the dictionary is assigned the same probability $\frac{1}{N}$, in which N is the total number of words that can be matched in **NormDic** after substitution of the Leet character.

Next, we present some detected top Leets and Leet patterns when including 3-grams in Table V, Table VI, and correspondingly, top Leets and patterns when not including 3-grams in Table VII, Table VIII. Some Leet transforms, such as "0" to "o", and Leet patterns like "zhang" to "shang", are prone to among the top Leets even under different condition. Thus,



(a) When including 3-grams

(b) When not including 3-grams

Fig. 2. The percentage of Leet pattern used in different datasets. “including 3-grams” means including n-grams where $n \geq 3$. “not including 3-grams” means including n-grams where $n > 3$.

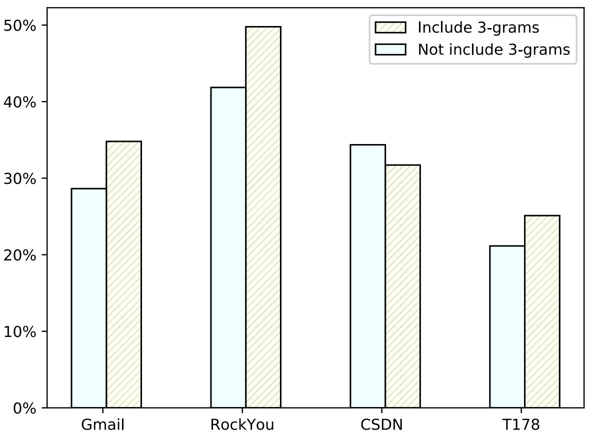


Fig. 3. All datasets’ Leet using proportion in **LeetDic**.

TABLE V

TOP LEETS AND CORRESPONDING NORMAL WORD WHEN INCLUDING 3-GRAMS

Gmail		**RockYou**		**CSDN**		**T178**	
Leet	**Normal**	**Leet**	**Normal**	**Leet**	**Normal**	**Leet**	**Normal**
3	e	0	o	0	o	4	a
0	o	3	e	3	e	3	e
4	a	1	i	4	a	1	i
1	i	4	a	1	i	0	o
1	t	1	t	5	s	z	s
p	o	1	l	2	r	5	s
1	l	p	o	1	t	12	r
5	s	2	r	z	s	1	t
2	r	v	u	8	b	2	r
s	z	@	a	9	p	j	y

TABLE VI

TOP LEET PATTERNS AND CORRESPONDING NORMAL WORD WHEN INCLUDING 3-GRAMS

Gmail		**RockYou**		**CSDN**		**T178**	
Leet	**Normal**	**Leet**	**Normal**	**Leet**	**Normal**	**Leet**	**Normal**
w3e	wee	luv	luu	200	zoo	w3e	wee
s@y	say	sha	zha	w3e	wee	we1	wet
sha	zha	200	zoo	zhang	shang	ver	fer
luv	luu	ver	fer	shi	zhi	g00d	good
passw0rd	password	no1	not	jing	ying	zhang	shang

they may act as the most frequently used Leet transformations in the majority of scenarios. Well-known transformations like several kinds of “password” (“passw0rd”, “p@ssw0rd”) are proved to be universally used. As a side note, we find that when we do not include 3-grams, both the total number of Leet and Leet pattern usage shows a decline. Likewise, the majority part of the Leet patterns found is 3-grams. Thus, to be comprehensive and thorough enough, we always factor 3-grams into the following study.

Interestingly, by using linear regression, we find that the distribution of real-life Leet patterns in user passwords obeys the Zipf’s law, which resides in natural languages and passwords [54]. For a password dataset D, the rank r of a Leet pattern and its frequency f_p follow the equation 2

$$f_p = \frac{C}{r^s} \tag{2}$$

where C and s are constants depending on the chosen dataset and $CovLen$. In order to better observe Zipf’s law, we plot the data on a log-log graph (base 10 in this work) in Figure 4, in which the axes being log(rank order) and log(frequency). Namely, $log(f_r)$ is linear with $log(r)$:

$$logf_r = logC - s \cdot logr \tag{3}$$

All of our datasets’ patterns have a high coefficient of determination (i.e., For Gmail, $R^2 = 0.953$; For RockYou, $R^2 = 0.974$; for CSDN, $R^2 = 0.968$; and for T178, $R^2 = 0.972$.), which indicates a remarkably sound fitting. Note that the Gmail dataset falls behind slightly in the fitting. This may due to bias of the raw dataset [21]: this dump may also contain about 2.5% yandex.ru addresses. However, as the Leet detection and cracking session does not rely on the

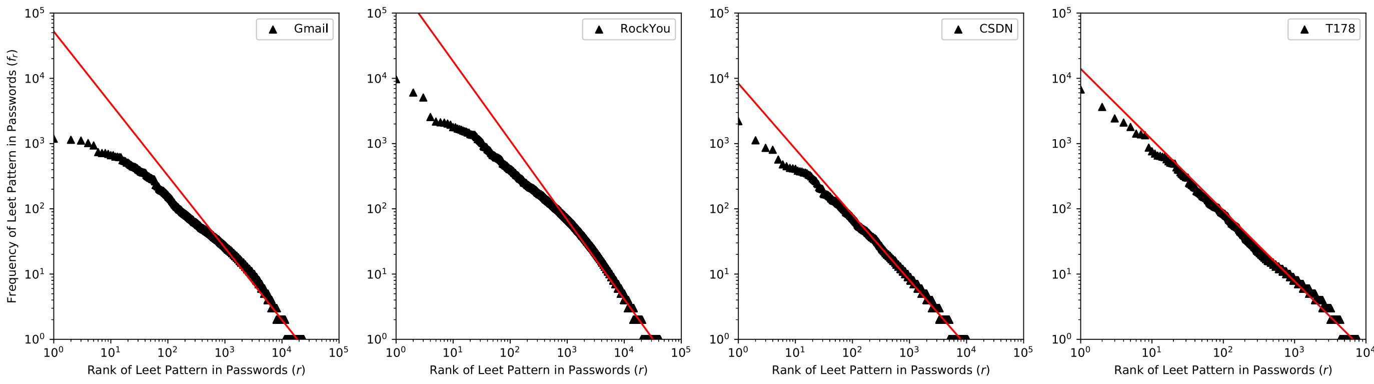


Fig. 4. Zipf's law in real-life Leet patterns in user passwords plotted on a log-log scale.

TABLE VII
TOP LEETS AND CORRESPONDING NORMAL WORD WHEN NOT INCLUDING 3-GRAMS

Gmail		RockYou		CSDN		T178	
Leet	**Normal**	**Leet**	**Normal**	**Leet**	**Normal**	**Leet**	**Normal**
0	o	0	o	0	o	z	s
3	e	3	e	z	s	4	a
4	a	1	l	4	a	5	s
1	l	4	a	5	s	3	e
1	i	1	t	3	e	s	z
5	s	1	i	2	r	0	o
1	t	2	r	s	z	12	r
2	r	z	s	1	l	1	t
@	a	5	s	1	t	2	r
z	s	p	o	9	g	1	i

TABLE VIII
TOP LEET PATTERNS AND CORRESPONDING NORMAL WORD WHEN NOT INCLUDING 3-GRAMS

Gmail		RockYou		CSDN		T178	
Leet	**Normal**	**Leet**	**Normal**	**Leet**	**Normal**	**Leet**	**Normal**
passw0rd	password	zhan	shan	zhang	shang	zhang	shang
shan	zhan	100p	loop	jing	ying	zong	song
play	olay	love2	lover	zhao	shao	dr4g	drag
p@ssw0rd	password	l0ve	love	zhou	shou	jing	ying
ana1	anal	s4me	same	p@ssw0rd	password	wi1l	will

distribution, the result about Leet transformations in password will not be affected.

C. Cracking Results

In Figure 5, we compare the performance of the original PCFG and PCFG with Leet transforming (Leet-PCFG) using the RockYou dataset as an example, for it is the largest dataset in this paper. As mentioned before, we use half of the dataset as the training set and the other half as the testing set.

To start with, we test the affection of β in the cracking session. We adopt three values of β, 0.01, 0.05 and 0.1. Given the different number of guesses, we compute the percentage of those cracked passwords in the entire password trial set. Figure 5(a) shows the hitting result of the primary PCFG and its replacing outcome in an offline attack. The cracking rate increases quickly because they always try high probability guesses first. At first, there is little difference between the two methods. The reason is that whatever replacement rate is, the high possible passwords that can be enacted as transforming material are not ample enough for Leet change to create more match passwords in the testing set. As the cracking session goes on, the possibility of generated guess tends to decrease, which leads to the slower gradient exaltation of the curve. Figure 5(d) shows that 5% of Leet replacement helps to crack 0.55% more passwords in the test set after about 1 billion times of attacks. Cracking with the same parameters leads to similar outcomes in other password sets. The accuracy improvements are 0.75% in Gmail, 0.48% in T178, and 0.56% in CSDN.

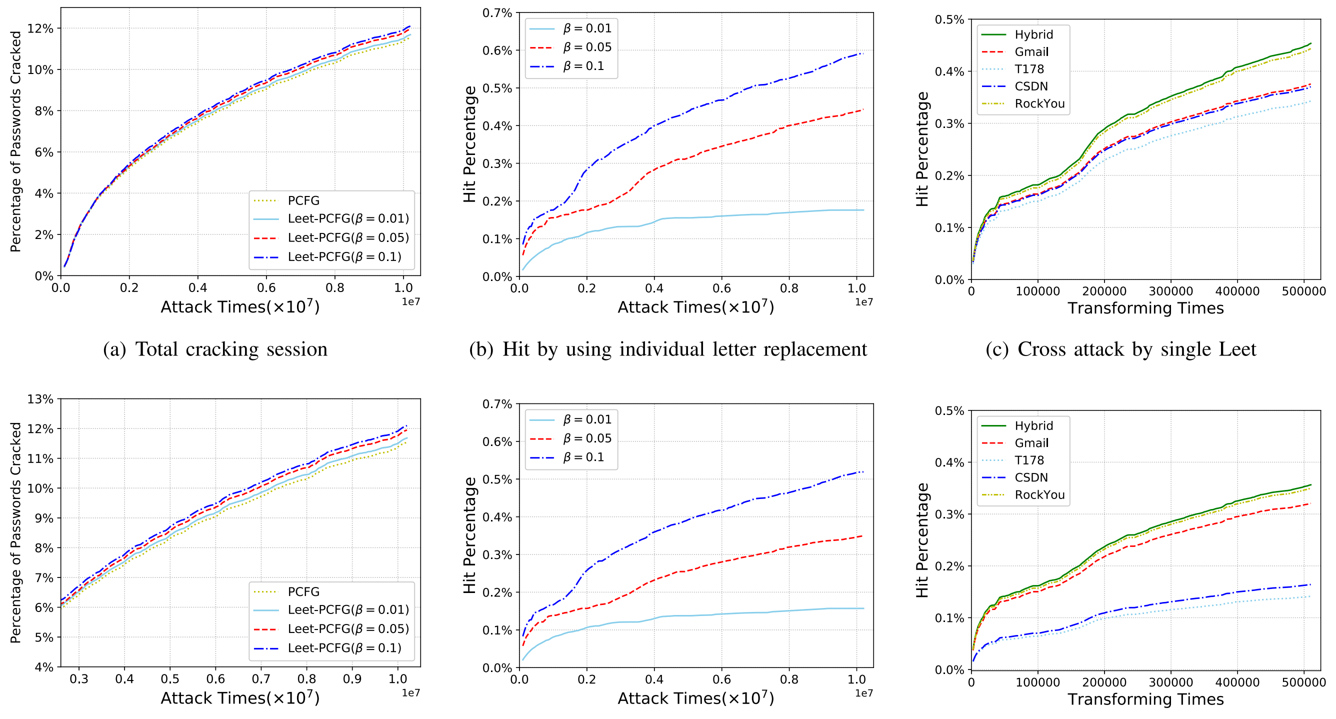
To investigate the attack performance of passwords generated by Leet transforming, we define hit percentage as the ratio of the size of $\{p_o\}$ to the number of training passwords. The hit percentage varies with attack times under different replacing rate β, and the results are shown in Figure 5(b) and 5(e). The hit percentage increases as β becomes large. However, as β increases, the replacing efficiency degrades. Therefore, we set β to 0.05 in later experiments to make a trade-off between the replacing efficiency and matching accuracy.

Here we notice two interesting facts. First, to achieve the same attack success rate, the PCFG model without considering leets requires more guesses. Thus, adding leets to existing attack model seems to increase its strength.

Second, the Leet using percentage varies in every dataset, which indicates that the user group's password setting strategy influences the overall characteristic of the website's password set. This is especially true for Leet patterns. A Leet pattern's using frequency could go up to the top list in one password set but down into the tail in another set. For example, the pattern "p@ssw0rd" in *T178* training set only appeared 18 times, but in the training set of *CSDN*, whose size is close to *T178*, "p@ssw0rd" was counted 274 times, more than 15 times as in *T178*.

D. Cross Attack

In previous experiments, we showed cracking results when attacking one set with Leets extracted from the same password set's training set. Such a condition guarantees consistency of password policies and user behaviors. However, this is true for some cases that we might not have enough resources to enable us to know about our target, or in lack of available sufficient



(a) Total cracking session (b) Hit by using individual letter replacement (c) Cross attack by single Leet

(d) Enlarged scale of cracking session (e) Hit by using pattern replacement (f) Cross attack by Leet pattern

Fig. 5. Cracking results. (a) shows PCFG vs. single Leet-PCFG with different β. (d) enlarges the scale of the tale. (b) and (e) compare hit percentage of transformed passwords in training set($\beta = 0.05$) under different replacement strategies (by single letter or pattern). (c) and (f) exhibit the hit percentage of transformed passwords in training set($\beta = 0.05$), also under different strategies.

training passwords. Consequently, a general Leet dictionary does good for all crack attempts.

To further establish the applicability of our findings, we synthesize all selected top Leets of the datasets to generate a general Leet dictionary. We multiply a Leet's rank and the size of the dataset it comes from to weight this Leet, then merge Leets from four datasets all together. Those among the top of the mixed Leet set are single out as part of the top Leets for all. The Leet patterns for all are elected with the same method. Later, we conduct a cross-crack test in *RockYou* password set using four groups of top Leets, including those from our real world dataset and the hybrid one (yet quite realistic). Figure 5(c) and 5(f) show the cracking results. The transforming times are the attempts ($\beta \cdot T$) made by Leet-PCFG, and here $\beta = 0.05$. Using the cracking result of RockYou Leets as the baseline, we notice that cracking by hybrid set shows the best result, although the improvement is small. We also find that English- and Chinese-based password datasets have distinctive effect on cross-cracking, especially when the cracking depends on Leet patterns (In Figure 5(f), set *RockYou* and set *Gmail* are more similar, while set *T178* and set *CSDN* are close). Since users can be grouped into a handful of behavioral clusters [42], the cause of this could be the variety in users of Internet services providers. Nevertheless, the hybrid of several datasets could rectify the shortcoming and make the result more surgical. In other words, it enlarges the total user sample pool. However, Leet patterns show less portability. We suspect that this is because those patterns have a more considerable correlation with particular user groups, which are harder to transplant.

As mentioned before, the pattern transformation is always included in the single Leet transformations, but it is more applicable in exact datasets. The phase of Leet pattern matching can derive more than 24,000 different patterns. Some Leet patterns appear in extremely high frequency, but in general, it has a long tail, as shown in Figure 4. However, in our attacking phase, some patterns with relatively lower regularity in the training set were found useful.

V. Survey

To properly evaluate the reliability of our algorithm in detecting Leets and further explore whether using Leets is acceptable for users, we design an online survey to collect users' reactions to those high-frequency Leet transformations.

A. Survey Design

We took various measures to improve the reliabilty of the survey. To ease the concerns that may lead to untrue results raised by [46], we randomized the sequence of questions and conducted the survey online. To prohibit multiple submissions from the same respondent, we set up a browser cookie examiner. To ensure the anonymity of our participants, we use a self-administered questionnaire.

We asked a series of questions listing passwords that contain possible Leet transformation dissected from our datasets. The sample passwords are randomly chosen from cracked passwords of each training set according to proportion. Parts of survey passwords includes high-frequency Leet patterns as well as top Leets, whereas the other only contains Leet

TABLE IX

Some Highly Approved Sample Passwords. Approving a Leet Password Means the Respondent Think Such Replacement Is Reasonable and Readable, so He or She Chose "Yes" Under This Question

Password	Leet letter	Original letter	Leet pattern	Approval Degree
g00dluck	0	O	g00d	86.76%
p@ssword	@	A	p@ss	84.11%
sh4dow	4	A	h4d	70.86%
w1ndows	1	I	w1nd	85.43%
2hangwei	2	Z	2han	74.17%

characters but not highly detected pattern. One example of the questions is "g00dluck" with a juxtaposed prompt, "[0, O]", to remark the pair of Leet and its w_o. In this case, Leet pattern is "g00d". Participants are asked to judge whether they think this password can be considered to contain a Leet or not in a short time.

In order to keep the survey form within a bearable length, we limit the number of questions to 60, covering 45 different Leet patterns and 19 top Leets. Participants were asked to answer from intuition but raptly to avoid fatigue. Typical participants would take about five minutes to complete it. We collected opinions from a cross-section of users, including students and professional staff at several universities as well as many other occupations. More than a half of participants have experience in the field of computer science, some even expert in web security. Thus, they are more likely to have knowledge of Leets. We also include lay people in the survey for the sake of the composition balance of respondents. Some lay people have never been exposed to any training of using or distinguishing Leets, even the notion itself. Likewise, they might not aware of Leets' importance in passwords as survey in [17] discovered. Thus, we start our survey by compendiously introducing of Leet transformation to help all participants understand the conception.

B. Responses and Summary

We received a total of 172 responses. On average, 65.28% of chosen passwords are considered to involve Leets. This portion is much higher than Figure 3 shows, which indicates the effectiveness of our filtering strategy. Some passwords as in Table IX are outstandingly and commonly approbated. We noticed that the majority of top Leets and Leet patterns agree with Table V and Table VI, which including 3-grams, but not with Table VII or Table VIII, which not including 3-grams. We surmise the reason lies at overfitting in the matching algorithm. Some subsequences or letters can be both regarded as Leet(s) or simply as ordinary character(s). To those who are not familiar with the usage and aim of Leets, such transformation might be ambiguous and not a necessity.

We find that users are inclined to accept or recognize a Leet when its shape is remarkably the same as the original letter in some cases, like [0, O]. It complies with the common sense of human nature. Besides, we notice that some Leet patterns with a relatively lower appearance in training set enjoy a higher acceptance rate. The phenomenon may due to the ambiguous

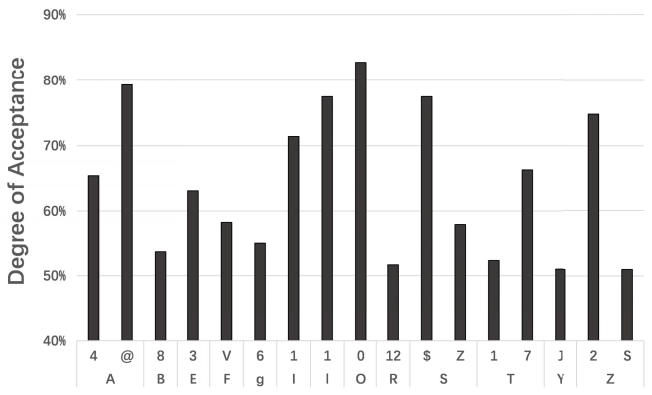


Fig. 6. Top voted Leets in the survey.

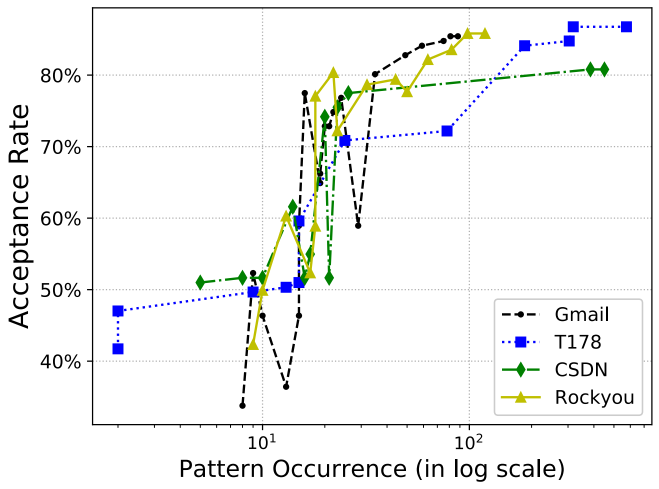


Fig. 7. Acceptance rate of patterns in the survey.

property of Leets. For example, both "shang" and "zhang" are a meaningful sequence in Chinese Pinyin, and "S" can be considered as a Leet of "Z". Furthermore, such transformation performs well in our experiments. All of the supported rates of voted Leets are summarized in Figure 6. Some of them are detected and matched many times in the dataset, but have low acceptance, like [J, Y] and [12, R]. We speculate that parts of the reason lie in the generality of participants.

We summarize the response of agreed Leet pattern in Figure 7. In general, more than half of the participants would accept a pattern if its overall occurrence in the original dataset is more than about 20 times. Such an outcome also supports our former assumption that adding patterns detected over around 20 times into a crack dictionary is a better choice to balance efficiency and accuracy. The relative consistency of base appearance times could act as a standard for researchers to determine a favorable selection strategy.

VI. Discussion

A. Scalability

As the workflow in Figure 1 shows, the "Leet matching" session and "guessing with Leet transformation" phase are apart from PCFG guess and attack. In other words, only the

"PCFG guess generation" phase is about attacking methods. The input and output of this phase are training password sets and guess results, respectively, which conform to mainstream methods like JtR [6] and neural networks. Likewise, the Leet dictionary is independent of the base method. Those Leet presentations and their using frequency could be combined with many related methods.

The Leet approach is likely to provide improvements over other algorithms ignoring Leets. For extensions of PCFG like Personal-PCFG [30], using Leets could be helpful as its basic strategy is still PCFG. For conceptually different algorithm such as approaches based on Markov models, Leets could also act as a supply. As although Markov model assigns possibilities to adjacent letters, it does not take the context of Leet into account.

B. Password Protection

We hope that our work brings benefits to researchers and system administrators by improving the understanding of Leet usage in passwords, including single Leets and Leet patterns, which may lead to more applicable measures of strengthening passwords.

Over the past fifteen years, numerous surveys and studies suggest that password reusing behavior is more prevalent than previously believed; users set the same or similar passwords across different websites (e.g., [13], [17], [19], [42], [55]). Also, studies like [53], [61] have found very low variation in the patterns that users leverage to modify their passwords when they modify their passwords at all. Considering the prevalence of user's transforming behavior, top Leets could be helpful to adversaries when the base attack dictionary is not comprehensive enough.

On the other hand, Leet pattern transformation is truly an efficient and memorable strategy to protect user passwords. According to Shay *et al.* [46], Wash *et al.* [56] and many other researchers, users tend to modify old and the most complicated passwords to create new ones. Even when they were forced to change, many users avoid creating completely new passwords. Using dictionary words and names with special characters attached to the beginning or end is still the most common strategies to create passwords. Consequently, the simple attempt to embrace some transform tricks could strengthen users' passwords by equipping little known Leets in single or pattern form.

C. Limitations

Leet has long been a quite instinctive choice even more than passwords. It is knotty to decide whether the user is using a Leet transformation or a word that coincidentally contains the letter sequence, especially for short patterns and niche expressions. For example, a fan of the band Green Day would claim that "tre3cool" (a member of the band) should not be interpreted as "treecool" or "tree" "cool", which makes perfect sense to those who do not know musician Tré Cool at all. In our survey, the ignorance of percentage of age, gender, language, and vocation distribution may lead to underestimating the problem. For example, as we mentioned before, "zhang" and "shang" are both meaningful to Chinese speakers, but not to users from other countries.

Moreover, we only considered the transformation of converting alpha to digits or special characters in our study. There might be some cases that users want to utilize other sorts of transformation, like from digits to letters. Nevertheless, as we conclude from our survey responses, such a pair of passwords could be somewhat paradoxical. As both of them can be seen as the password users initially want to use, it is of great difficulty to assert those kinds of transformation.

D. Ethical Considerations

Utilizing datasets leaked from data breaches has been a mainstream method on password studies. However, we fully realize that studying leaked passwords brings in ethical concerns. Each dataset in our study is stored and used with great attention. All data are only used for researching purpose. We will not expose any password or use this information in any way other than for research use.

For our user study, we took careful steps to address privacy issues regarding collecting and analyzing the data. Our questionnaire is anonymous, and respondents don't have to provide any information about themselves.

VII. Conclusion and Future Work

In this work, we conduct a comprehensive quantitative study on how Leet transformation resides in human-chosen passwords.

To the best of our knowledge, this is the first work to define and analyze the special transforming phenomenon in passwords systematically. We develop a Leet detecting method, and show its validity by cracking experiments and user surveys. Though the entire Leet transformation vocabulary is extensive, only around 30% of them are detected in real password sets. Leveraging two forms of Leets in password attacks performs similar accuracy, but using Leet patterns requires more attempt times. Our experiments prove that single Leet dictionaries extracted from different datasets could be synthesized to a more robust and easy-to-use universal dictionary, whereas Leet patterns are more case-based but offer higher efficiency. The recognizability of most of the detected Leet patterns and high-frequency Leet letters is proved by our user survey. In addition, this paper contributes one of the currently most comprehensive corpora of human-chosen Leet transformation dictionary, which improves on regular cracking models as a bonus. Leet transformation could also be accepted as a welcome and mnemonic method to enhance password strength, for it is simple and straightforward. The conclusion drawn from the research provides new perspectives for password analysis. According to the results of the paper, users always hold diverse attitudes towards Leets in passwords since subjective factors can be diversified.

Future work could be focused on to what extent people incorporating special transformation in passwords influences the password strength meters. The question whether digits and special characters could be transformed by users still remains. If so, their effect in password cracking, strength measurement, and password reusing are also interesting topics to be explored in future work.

REFERENCES

[1] *Hashcat Advanced Password Recovery*. Accessed: Mar. 5, 2019. [Online]. Available: https://hashcat.net/hashcat/

[2] *Leetspeak*. Accessed: Jan. 10, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Leet

[3] *Leet vocabulary*. Accessed: Jan. 10, 2019. [Online]. Available: https://zh.wikipedia.org/wiki/Leet

[4] *PCFG Cracker*. Accessed: Feb. 8, 2019. [Online]. Available: https://github.com/lakiw/pcfg_cracker

[5] (2007). *British National Corpus*. [Online]. Available: http://www.natcorp.ox.ac.uk/

[6] (2013). *John the Ripper Password Cracker*. [Online]. Available: https://www.openwall.com/john/

[7] A. Adams and M. A. Sasse, "Users are not the enemy," *Commun. ACM*, vol. 42, no. 12, pp. 41–46, 1999.

[8] H. Aoyama and J. Constable, "Word length frequency and distribution in English: Part I. Prose," *Literary Linguistic Comput.*, vol. 14, no. 3, pp. 339–358, Sep. 1999.

[9] J. Blocki, B. Harsha, and S. Zhou, "On the economics of offline password cracking," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 853–871.

[10] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 538–552.

[11] J. Bonneau, S. Preibusch, and R. Anderson, "A birthday present every eleven wallets? The security of customer-chosen banking pins," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2012, pp. 25–40.

[12] S. Boztas, "Entropies, guessing, and cryptography," Dept. Math., Roy. Melbourne Inst. Technol., Tech. Rep. 6, vol. 6, 1999, pp. 2–3.

[13] A. S. Brown, E. Bracken, S. Zoccoli, and K. Douglas, "Generating and remembering passwords," *Appl. Cognit. Psychol.*, vol. 18, no. 6, pp. 641–651, Sep. 2004.

[14] C. Cachin, "Entropy measures and unconditional security in cryptography," Ph.D. dissertation, ETH Zürich, Zürich, Switzerland, 1997.

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

[16] C. Castelluccia, A. Chaabane, M. Dürmuth, and D. Perito, "When privacy meets security: Leveraging personal information for password cracking," 2013, *arXiv:1304.6584*. [Online]. Available: http://arxiv.org/abs/1304.6584

[17] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled Web of password reuse," in *Proc. 21st Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2014, pp. 23–26.

[18] X. De. C. D. Carnavalet and M. Mannan, "From very weak to very strong: Analyzing password-strength meters," in *Proc. 21st Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2014, pp. 1–19.

[19] D. Florencio and C. Herley, "A large-scale study of Web password habits," in *Proc. 16th Int. Conf. World Wide Web (WWW)*, 2007, pp. 657–666.

[20] S. Gaw and E. W. Felten, "Password management strategies for online accounts," in *Proc. 2nd Symp. Usable Privacy Secur. (SOUPS)*, 2006, pp. 44–55.

[21] M. Golla, B. Beuscher, and M. Dürmuth, "On the security of cracking-resistant password vaults," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1230–1241.

[22] M. Golla and M. Dürmuth, "On the accuracy of password strength meters," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1567–1582.

[23] W. Han, Z. Li, L. Yuan, and W. Xu, "Regional patterns and vulnerability analysis of Chinese Web passwords," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 258–272, Feb. 2016.

[24] A. Hanamsagar, S. S. Woo, C. Kanich, and J. Mirkovic, "Leveraging semantic transformation to investigate password habits and their causes," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, Apr. 2018, p. 570.

[25] B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz, "PassGAN: A deep learning approach for password guessing," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Berlin, Germany: Springer, 2019, pp. 217–237.

[26] S. Houshmand, S. Aggarwal, and R. Flood, "Next gen PCFG password cracking," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1776–1791, Aug. 2015.

[27] P. G. Kelley *et al.*, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 523–537.

[28] J. Kiesel, B. Stein, and S. Lucks, "A large-scale analysis of the mnemonic password advice," in *Proc. 24th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2017, pp. 1–13.

[29] S. Komanduri *et al.*, "Of passwords and people: Measuring the effect of password-composition policies," in *Proc. Annu. Conf. Hum. Factors Comput. Syst. (CHI)*, 2011, pp. 2595–2604.

[30] Y. Li, H. Wang, and K. Sun, "A study of personal information in human-chosen passwords and its security implications," in *Proc. IEEE INFOCOM-35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[31] Z. Li, W. Han, and W. Xu, "A large-scale empirical analysis of Chinese Web passwords," in *Proc. 23rd USENIX Secur. Symp. (USENIX Secur.)*, 2014, pp. 559–574.

[32] Y. Liu *et al.*, "GENPass: A general deep learning model for password guessing with PCFG rules and adversarial generation," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[33] T. Lundberg, "Comparison of automated password guessing strategies," M.S. thesis, Linköping Univ., Linköping, Sweden, 2019.

[34] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 689–704.

[35] D. Malone and K. Maher, "Investigating the distribution of password choices," in *Proc. 21st Int. Conf. World Wide Web (WWW)*, 2012, pp. 301–310.

[36] M. L. Mazurek *et al.*, "Measuring password guessability for an entire university," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 173–186.

[37] W. Melicher *et al.*, "Fast, lean, and accurate: Modeling password guessability using neural networks," in *Proc. 25th USENIX Secur. Symp. (USENIX Secur.)*, 2016, pp. 175–191.

[38] G. A. Miller, E. B. Newman, and E. A. Friedman, "Length-frequency statistics for written English," *Inf. Control*, vol. 1, no. 4, pp. 370–389, Dec. 1958.

[39] R. Morris and K. Thompson, "Password security: A case history," *Commun. ACM*, vol. 22, no. 11, pp. 594–597, Nov. 1979.

[40] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proc. 12th ACM Conf. Comput. Commun. Secur. (CCS)*, 2005, pp. 364–372.

[41] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2003, pp. 617–630.

[42] S. Pearman *et al.*, "Let's go in for a closer look: Observing passwords in their natural habitat," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.* New York, NY, USA: ACM, Oct. 2017, pp. 295–310.

[43] J. O. Pliam, "On the incomparability of entropy and marginal guesswork in brute-force attacks," in *Proc. Int. Conf. Cryptol. India*. Berlin, Germany: Springer, 2000, pp. 67–79.

[44] D. Schweitzer, J. Boleng, C. Hughes, and L. Murphy, "Visualizing keyboard pattern passwords," *Inf. Visualizat.*, vol. 10, no. 2, pp. 127–133, Apr. 2011.

[45] R. Shay *et al.*, "Designing password policies for strength and usability," *ACM Trans. Inf. Syst. Secur.*, vol. 18, no. 4, pp. 13-1–13-34, 2016.

[46] R. Shay *et al.*, "Encountering stronger password requirements: User attitudes and behaviors," in *Proc. 6th Symp. Usable Privacy Secur. (SOUPS)*, 2010, pp. 1–20.

[47] B. Ur, "Supporting password-security decisions with data," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2018.

[48] B. Ur, J. Bees, S. M. Segreti, L. Bauer, N. Christin, and L. F. Cranor, "Do users' perceptions of password security match reality?" in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, May 2016, pp. 3748–3760.

[49] B. Ur *et al.*, "'I Added'!'At the end to make it secure': Observing password creation in the lab," in *Proc. 11th Symp. Usable Privacy Secur. (SOUP)*, 2015, pp. 123–140.

[50] B. Ur *et al.*, "Measuring real-world accuracies and biases in modeling password guessability," in *Proc. 24th USENIX Secur. Symp. (USENIX Secur.)*, 2015, pp. 463–481.

[51] R. Veras, C. Collins, and J. Thorpe, "On semantic patterns of passwords and their security impact," in *Proc. 21st Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2014, pp. 1–16.

[52] C. Wang, S. T. K. Jan, H. Hu, D. Bossart, and G. Wang, "The next domino to fall: Empirical analysis of user passwords across online services," in *Proc. 8th ACM Conf. Data Appl. Secur. Privacy*, Mar. 2018, pp. 196–203.

[53] C. Wang, S. T. K. Jan, H. Hu, and G. Wang, "Empirical analysis of password reuse and modification across online service," 2017, *arXiv:1706.01939*. [Online]. Available: http://arxiv.org/abs/1706.01939

[54] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2776–2791, Nov. 2017.

[55] K. C. Wang and M. K. Reiter, "How to end password reuse on the Web," 2018, *arXiv:1805.00566*. [Online]. Available: http://arxiv.org/abs/1805.00566

[56] R. Wash, E. Rader, R. Berman, and Z. Wellmer, "Understanding password choices: How frequently entered passwords are re-used across websites," in *Proc. 12th Symp. Usable Privacy Secur. (SOUP)*, 2016, pp. 175–188.

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

[58] R. Yampolskiy, "Analyzing user password selection behavior for reduction of password space," in *Proc. 40th Annu. Int. Carnahan Conf. Secur. Technol.*, Oct. 2006, pp. 109–115.

[59] S. Yang, S. Ji, and R. Beyah, "DPPG: A dynamic password policy generation system," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 545–558, Mar. 2018.

[60] J. Zeng, J. Duan, and C. Wu, "Empirical study on lexical sentiment in passwords from Chinese websites," *Comput. Secur.*, vol. 80, pp. 200–210, Jan. 2019.

[61] Y. Zhang, F. Monrose, and M. K. Reiter, "The security of modern password expiration: An algorithmic framework and empirical analysis," in *Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS)*, 2010, pp. 176–186.

Wanda Li received the B.S. degree (Hons.) from the School of Computer Science, Fudan University. She is currently a graduate student of data science and information technology with the Tsinghua-Berkeley Shenzhen Institute (TBSI). Her research interests include data mining, machine learning, and user behavior analysis.

Jianping Zeng received the Ph.D. degree from Xiamen University in 2006. He is currently an Associate Professor with the School of Computer Science, Fudan University. He has published over 70 research papers in refereed journals and conferences in the areas of network security and artificial intelligence, and has authored two books. He holds seven patents. His research interests include big data security, AI security, and machine learning for social media analysis.