Special Characters Usage and Its Effect on Password Security

Daojing He, *Member, IEEE*, Zhiyong Liu, Shanshan Zhu, Sammy Chan, *Senior Member, IEEE*, and Mohsen Guizani, *Fellow, IEEE*

***Abstract*—Continuously preventing weak password attacks is one of the most important initiatives to secure IoT and smart contract platforms. Despite their significance as crucial components of passwords, special character segments have been overlooked. This study systematically investigates the basic characteristics and semantic patterns of special character segments. We assess the efficacy of special character segment characteristics in cracking trials through assimilation into the latest probabilistic context-free grammar (PCFG$_{v4}$) method for password cracking by updating the preterminal structure or performing special character segment transformation. Experimental findings demonstrate that a mere 6% transformation rate improves the cracking rate by 3.72% under the optimal assimilation combination. Our investigation reveals that the current password creation policies of mainstream IoT platforms and smart contract wallets overestimate the strength of passwords with special characters. To enhance their passwords, users can employ low-frequency special character semantic strings. For IoT platforms or smart contract wallets, the use of blacklist constructed from special character segment characteristics can effectively mitigate the risk of overestimating the strength of passwords with special characters.**

***Index Terms*—Internet of Things (IoT) platforms security, password analysis, password protection, smart contract wallet security, weak password attack.**

Manuscript received 16 December 2023; revised 22 January 2024; accepted 5 February 2024. Date of publication 19 February 2024; date of current version 23 May 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62376074; in part by the National Key Research and Development Program of China under Grant 2021YFB2700900; in part by the Shenzhen Science and Technology Program under Grant KCXST20221021111404010, Grant JSGG20220831103400002, Grant JSGGKQTD20221101115655027, Grant KJZD20230923114405011, and Grant SGDX20230116091244004; in part by the Fok Ying Tung Education Foundation of China under Grant 171058; and in part by the University Grants Committee of the Hong Kong Special Administrative Region, China, under Project CityU 11201421. *(Corresponding author: Daojing He.)*

Daojing He is with the School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China, and also with the School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China (e-mail: hedaojinghit@163.com).

Zhiyong Liu is with the School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China.

Shanshan Zhu is with the School of Economics and Management, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China (e-mail: 1650791729@qq.com).

Sammy Chan is with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong (e-mail: eeschan@cityu.edu.hk).

Mohsen Guizani is with the Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE (e-mail: mguizani@ieee.org).

Digital Object Identifier 10.1109/JIOT.2024.3367323

I. Introduction

THE LARGE number of devices and users in an Internet of Things (IoT) platform requires an efficient and lightweight authentication technique to ensure that only legitimate devices or users can access the system. In this case, password-based authentication schemes come into play [1]. Smart contract platforms are a viable solution to the current skill shortage problem in the IoT. The solution enables the control of user contracting behavior by means of enabling a contract-based platform [2]. Users can interact with smart contracts and manage their decentralized applications and digital assets by using a smart contract wallet [3]. However, the security of smart contract wallets is highly dependent on the security of the user's password.

Currently, the majority of IoT platforms and smart contract wallets assist their users in comprehending their password strength through rigorous password creation policies. Several studies indicate that the use of password creation policies can effectively encourage users to select stronger passwords [5], [6]. Yet, overly strict password creation policies that mandate special character types may not always lead to the creation of robust passwords. Password creation policies can sometimes potentially misinform users and lead them to create passwords that they thought are secure but are actually not [5]. As to be discussed later in Section VI-A, password creation policies of mainstream IoT platforms and smart contract wallets have a tendency to overestimate the strength of passwords with special characters. Unfortunately, users frequently change their passwords across various online services to those containing special characters [7], [8]. This practice, concentrates the majority of password variations in a confined part of the password space, making brute-force or dictionary attacks feasible. Consequently, passwords with special characters under IoT or smart contract platforms are often not entirely secure, thereby presenting a potential security hazard to the platforms. To accurately assess the strength of passwords with special characters, it is necessary to understand the exact methods that users use when creating strings of letters, digits or special characters. By understanding these methods, IoT and smart contract platforms administrators can optimize password creation policies to mitigate potential hazards, and users can generate stronger passwords by consciously changing poor password creation habits.

Previous studies investigated the composition of user passwords [9], common elements, such as keyboard sequences [10], date sequences, name sequences [11], and frequently used sequences (e.g., "password" and "123456" [12]), become evident. They often manifest in passwords as letter or digit segments in everyday life. In this article, we examine a frequently utilized component of passwords, i.e., special character segments. From an appraisal of current literature, we establish that present password cracking methods have insufficiently considered the incorporation of special character segments in passwords. While some studies mentioned the common distribution of special characters [13], and Leets about special characters [14], [15], [16], there is no comprehensive analysis of special character segment characteristics, and in most cases only a few characteristics are considered.

Our aim is to discern typical and favored basic characteristics and semantic patterns in special character segments and to show their impact on the strength of passwords with special characters, i.e., whether the combination of the main characteristics found with existing attack methods can improve the cracking performance. In this study, we use the PCFG$_{v4}$ model as the base model for password cracking experiments. It is the latest version of the well-known probabilistic context-free grammar (PCFG) cracking framework [17], [18], [19]. Rule-based attacks may outperform other guessing methods when only a few guesses are allowed [20]. Although recurrent neural network (RNN)-based guessing methods [21], [22] have recently gained popularity as a state-of-the-art approach, their interpretability is too unclear for our purposes.

Our contributions can be summarized as follows.

1) We carry out original and comprehensive investigation of the special character segments in password creation, which identifies the fundamental characteristics and semantic structures of special character segments in passwords with special characters. We propose a method for special character semantic string detection that intelligently extracts all strings in a data set that match a semantic pattern.
2) We suggest two methods to enhance PCFG$_{v4}$ attacks. In contrast to the original PCFG$_{v4}$, the improved PCFG$_{v4}$ includes characteristic attacks targeting special segments in passwords. These two methods of improvement operate independently of each other and can increase the success rate of PCFG$_{v4}$ in cracking passwords with special characters through superposition optimization.
3) We evaluate the reasonableness of password creation policies of mainstream IoT platforms and smart contract wallets in guiding users to create special character segments, and propose a method to improve the accuracy of password creation policies in assessing the strength of passwords with special characters.

The remainder of this article is organized as follows. Section II covers related research. In Section III, an overview of the password data sets is presented, along with the introduction of a semantic pattern detection method and an analysis of the fundamental characteristics of special character segments. In Section IV, we present a detailed explanation of the methodology for enhancing PCFG$_{v4}$ with semantic dictionaries. In Section V, we evaluate the performance of our proposed password attack methods. Section VI explores the precise contributions and limitations of our research, and in Section VII, we conclude our work.

II. Related Work

A. Password Analysis

For many years, the implementation of text-based passwords has been the primary policy for Internet authentication. However, although passwords are commonly used in Internet services, their security is still a thorny issue. Due to the limited capacity of the human mind, users tend to choose simple and memorable passwords, which are typically easy to crack [23]. Conversely, random and hard-to-crack passwords are often difficult to remember [35], [37]. Therefore, online services aim to strike a balance between password accessibility and security through the implementation of password creation policies [24] and blacklists [25], [26]. Extensive research has been conducted to comprehend the password-creation behaviors of users, evaluate the security of their passwords, and fortify their passwords to enhance their strength. These findings provide a crucial foundation to enhance password policies and broaden blacklists. Researchers have discovered several promising characteristics in users' passwords.

Wang et al. [13] conducted an investigation on over 100 million actual Chinese passwords and highlighted that Chinese passwords significantly vary from those in other languages in their utilization of name segments and date-of-birth segments. On the other hand, Alsabah et al. [27] analyzed multiple data sets and demonstrated that 9% of users still willingly used special characters without prompting and tended to select frequently used special characters. Replacing "a" with "@" and replacing "s" with "$" were recommended policies by Gerlitz et al. [28] after analyzing various password creating policies. Analyses including human intrinsic factors, such as habits [29], overall inspiration [30], semantic patterns [31], and security awareness [32], were also conducted. Yang et al. [33] examined the effectiveness of the most commonly used password strength meters on the Internet. Their findings showed that these password strength meters exhibit a high degree of ambiguity in evaluating special character segment strength and do not consider the impact of special character segment characteristics on password strength.

Most previous research has centered on the characteristics of letter and digit segments within passwords. While a few studies have touched upon the common characteristics of special character segments, there has been limited elaboration on these details. Special character segment characteristics are one of the basic components of password characteristics, and their study can provide clues for the study of password analysis, cracking and strength measurement. Unfortunately, current methods for cracking passwords do not give sufficient attention to the utilization of special character segment characteristics within passwords. In this work, we will comprehensively analyze the use of special character segments in passwords and reveal the scope and extent of their influence on password strength.

B. Password Attack

Discussions surrounding password cracking methods have been present since the inception of passwords. For attackers, brute force and dictionary attacks remain the fundamental options. When considering common password characteristics, dictionary attacks can be particularly effective [23]. Password cracking methods that use dictionaries, such as John the Ripper (JtR) [34] and HashCat [14], typically employ dictionary entries to generate guesses, which are then modified in a conventional manner. Such methods have limitations in terms of the number of guesses and time required. Furthermore, probabilistic models have been used to develop different approaches. Narayanan and Shmatikov [35] devised a probabilistic method with Markov chains, while Castelluccia et al. [36] presented OMEN, built upon the model introduced in [35], to enhance cracking proficiency. Weir et al. [37] proposed PCFG to train a password set that is partially public, allowing for the generation of guesses based on the resulting grammar. Probabilistic threshold graphs have proven to be superior to guess number graphs [38] and consequently have been used as the foundation for several studies, including [18]. These studies have provided diverse perspectives and enriched previous research [10], [39], [40].

Numerous researchers have utilized neural networks for password creation or assessing their susceptibility to security breaches [20]. Melicher et al. [21] investigated the use of RNN for generating guess passwords and showed that they tend to produce passwords more efficiently compared to advanced methods like Markov models [35] and probabilistic contexts [37]. However, RNNs prioritize generating a greater number of password guesses while hindering details, which goes against conventional thinking. While they may have superior performance, their usefulness may be restricted due to the requirement of additional guessing attempts [20].

C. Password Pattern

Incorporating characteristic results acquired from password analysis into attack models can effectively enhance guessing performance. Houshmand et al. [10] methodically integrated keyboard patterns and multiword patterns into the original PCFG model. Li et al. [41] expanded the PCFG model to enhance its semantic richness by incorporating personalized characteristics like name, email address and date of birth as new structures within passwords. Reference [42] updated the latest version of the PCFG (PCFG$_{v4}$). PCFG$_{v4}$ added to the basic structure of the original PCFG the additions of K (keyboard sequences), Y (year sequences), and E (email address sequences) sections and differentiates between upper and lower case letters to provide better cracking rates than the original PCFG.

Despite the addition of special characters, such as ".", "!", and "*", or the transformation of "password" to "P@$$word," the most typical practices for using special characters have become standard examples in special character segments. Nevertheless, few studies have adequately explored the utilization patterns of special character segments by users. The use of special character segments in passwords presents a pragmatic issue that deserves measurement, as they are a necessary characteristic of creating special character segments [15]. There is a lack of empirical research into existing special character segments, which inspires the need for more extensive investigation.

III. Analysis of Special Character Segments Characteristics

A. Data Sets

Some prior password research has employed leaked passwordsets in their investigations as the number of passwords provides adequate data for statistical analysis. Analogously, we utilized four large-scale password data sets that have been leaked in recent years. All of our data sources have been made public at least once, with no accuracy issues raised by affected websites. We merely obtained passwords but not the corresponding account information. It would be considered unethical to verify the authenticity of the data using credentials without appropriate consent. The details of the password sets are detailed below.

China Software Developers Network (CSDN) is a large online forum in China, similar to GeeksforGeeks. Its users are mainly software developers, electronic engineers and computer science students. Youku, a China-based online video platform that offers a range of terminal services, experienced a password data set breach from an attack on their database in 2016. Similarly, Myspace.com, a social networking website providing various services globally, also suffered a password data set breach during that year. Zoosk, the largest social dating platform in the world that primarily caters to single individuals, experienced a data breach in 2018. Hence, by choosing passwords acquired from multiple websites with diverse characteristics and user demographics, these data sets furnish sufficient proof for our research on the employment of special character segment in passwords.

We excluded passwords containing special characters beyond the ASCII character set. Moreover, we omitted passwords with a special string length exceeding ten characters and passwords exceeding 30 characters in length as they appear to be useless and most likely system-generated or junk information [43]. The information of the cleaned data is shown as follows.

1) *CSDN:* 6 428 632 passwords, including 208 609 passwords with special characters. The percentage of passwords with special characters is 3.24%.
2) *Youku:* 47 622 107 passwords, including 2 870 066 passwords with special characters. The percentage of passwords with special characters is 6.03%.
3) *Myspace:* 114 735 301 passwords, including 13 128 817 passwords with special characters. The percentage of passwords with special characters is 11.44%.
4) *Zoosk:* 29 012 757 passwords, including 3 274 877 passwords with specials. The percentage of passwords with special characters is 11.29%.

It appears that the percentage of passwords with special characters is considerably greater in the Myspace and Zoosk data collections compared to the CSDN and Youku collections.

TABLE I
DISTRIBUTION OF THE QUANTITY AND POSITION OF SPECIAL SEGMENTS IN PASSWORDS WITH SPECIAL CHARACTERS

Dataset	Quantity distribution		Position distribution					
	Single	Multiple	Unique position			Multiple positions		
			Head	Middle	Tail	Head-tail symmetry	Fixed format	Other
CSDN	**182,984(87.71%)**	25,625(12.29%)	10,289(5.63%)	**92,755(50.67%)**	79,940(43.70%)	3,845(15.01%)	**8,046(31.40%)**	13,734(53.59%)
Youku	**2,607,206(90.84%)**	262,860(9.16%)	317,036(12.16%)	**1,291,871(49.55%)**	998,299(38.29%)	31,885(12.13%)	**65,084(24.76%)**	165,891(63.11%)
Myspace	**11,540,844(87.95%)**	1,587,973(12.05%)	668,017(5.79%)	**5,863,902(50.81%)**	5,008,925(43.40%)	**261,539(16.47%)**	140,535(8.85%)	1,185,899(74.68%)
Zoosk	**2,573,522(78.58%)**	701,355(21.42%)	122,242(4.75%)	**1,404,113(54.56%)**	1,047,167(40.69%)	**111,725(15.93%)**	81,076(11.56%)	508,554(72.51%)

This suggests that English users tend to prefer using special character segments in their passwords more frequently than Chinese users. To better analyze the basic characteristics of special character segments within the data set, we chose all passwords with special characters in the original data set as the data set used for the analysis and experiments in this article. Then, the following characteristics were considered.

1) *Quantity of Special Character Segments:* We investigated cases where the quantity of occurrences of special character segments is single or multiple.
2) *Position of Special Character Segments:* We defined the positions of segments in password as unique position or multiple positions.
3) *Special Characters Frequency:* Frequency of top-10 special characters in each data set.
4) *Special Character Segment Lengths Frequency:* We arranged the top-10 segment lengths (referred to as "the lengths of segments") in descending order of frequency. Then, we defined the top-5 segment lengths as the popular lengths, as their frequency sum exceeds 98%.

According to the above four characteristics, we have analyzed the data sets in detail and the results are discussed below.

B. Basic Characteristics of Special Character Segments

We defined a continuous special string as one special segment. In the following text, if not specifically emphasized, the passwords refer to passwords with special characters, and the special segments refer to special character segments. There are four basic characteristics of special segments.

1) Quantity of Special Segments: Table I shows that most passwords in the four data sets possess only one special segment, with less than 15% of passwords containing multiple special segments. This suggests that users typically opt for a single special segment when considering their password composition. One possible explanation is that entering and recalling multiple special segments can be challenging.

2) Position of Special Segments: Table I indicates that single special segments are typically located in the middle or at the tail of passwords in the four data sets, while such segments are hardly ever found at the head of passwords. Furthermore, our analysis of the distribution of multiple special segments in passwords uncovered some notable characteristics. For instance, approximately 15% of the passwords with multiple special segments in each data set are symmetrically distributed regarding the head and tail positions of these segments. In the Chinese data set, approximately 30% of passwords with numerous special segments have a consistent arrangement of these segments in designated positions, such as date format, URL format, email address, and IP address format. In contrast, this pattern is not evident in the English data set. The present results indicate the presence of notable characteristics concerning the allocation of special segments in passwords. These are primarily influenced by the quantity of special segments incorporated in passwords and the linguistic background.

TABLE II
FREQUENCY OF SPECIAL CHARACTERS IN FOUR DATA SETS

Rank	Special character frequency			
	CSDN	Youku	Myspace	Zoosk
1	**.(28.67%)**	**.(43.97%)**	**.(20.69%)**	**!(17.07%)**
2	**@(17.24%)**	**-(19.08%)**	**!(17.61%)**	**.(14.88%)**
3	***(9.01%)**	@(7.78%)	@(7.60%)	**@(12.95%)**
4	!(8.97%)	*(5.71%)	-(7.40%)	**?(11.89%)**
5	-(5.76%)	!(4.83%)	*(6.99%)	\O(8.57%)
6	#(5.03%)	+(4.35%)	#(6.46%)	-(7.97%)
7	+(4.76%)	,(3.03%)	$(5.50%)	$(7.01%)
8	/(2.75%)	/(2.44%)	&(4.32%)	+(5.15%)
9	$(2.66%)	?(2.10%)	'(3.03%)	*(3.25%)
10	?(2.37%)	#(1.49%)	%(2.63%)	#(2.36%)
% of top-10	**87.22%**	**94.78%**	**82.23%**	**91.10%**

3) Special Characters Frequency: We considered the incidence of 31 distinct special characters that can be employed in passwords. Based on probability, we anticipated that the frequency of these characters should be approximately equivalent. Nevertheless, in actual password usage, this projection is inaccurate. Table II illustrates that the full stop symbol (".") has a considerably high frequency in all four data sets, with an average frequency of 36% in the Chinese data set and 17% in the English data set. This is significantly higher than the random distribution level. Furthermore, our findings reveal that users of different languages exhibit distinct character preferences. For instance, Chinese users prioritize the use of the full stop symbol ("."), followed by the at symbol ("@"), hyphen ("-"), and other commonly used symbols. In contrast, English users tend to use a combination of the period and exclamation symbol ("!") and also favor the at symbol ("@") and other frequently used symbols. Furthermore, the sum of the frequencies of the top-3 special characters in all four data sets accounts for approximately 50%. If we extend the frequency summation to the top-10 special characters, it will increase to approximately 85%. One hypothesis is that Internet users favor special characters that are familiar

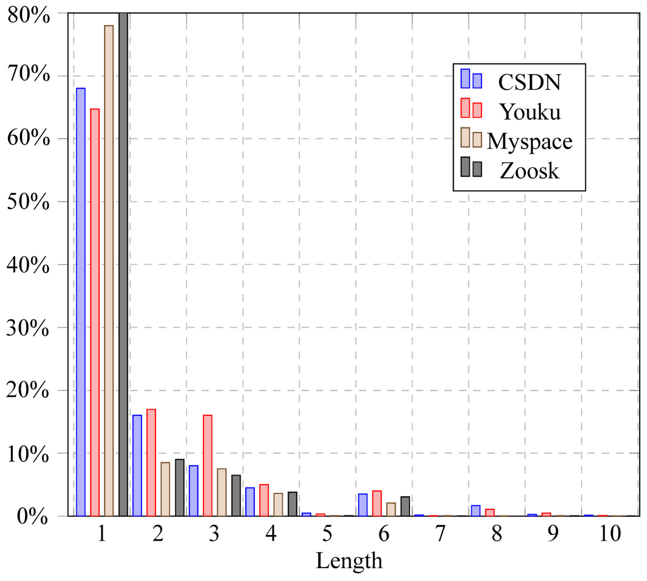

Fig. 1. Length distribution of special segments.

to them, particularly those that are ubiquitous in their day-to-day activities (such as ".", "@", and "*"), as a means of resolving the challenge of easily remembering specific segments. These discoveries propose that there are unique characteristics concerning Internet users' selection of special characters, and that such traits are, to some extent, shaped by their linguistic background.

4) Special Segment Length Frequency: As illustrated in Fig. 1, it is evident that the proportion of segment length 1 in the four data sets is significantly greater than that of other segment lengths, with a more pronounced trend seen in the English data set. In addition to segment length 1, the combined proportion of segment lengths 2, 3, 4, and 6 in the four data sets should not be disregarded, as their cumulative sum ranges from 19% to 31%. Notably, the English data set appears to favor the former, whereas the Chinese data set favors the latter. Therefore, it is crucial to consider these segment lengths when analyzing the data sets. It is noteworthy that across all four data sets, the combined frequency of the original three segment lengths totals over 92%. Furthermore, the proportion of these segment lengths shows an exponential decline. These outcomes indicate that Internet users predominantly utilize a single special character, followed by special segments of length 2 or 3, with a corresponding drop-off in probability as length increases. In the following section, we refer to these five segment lengths mentioned above as the popular lengths for special segments.

C. Semantic Characteristics of Special Segments in Passwords

1) Semantic Patterns of Special Segments: In this section, we will present a method for extracting distinct semantic strings. Using this approach, we will identify and record all conceivable semantic strings and their corresponding semantic tags for each popular length segment. We define the possible semantic patterns of special segments as the following four semantic tags.

1) *Reuse Sequence:* In this case, we consider two reuse patterns. The first is the pattern of reusing only a single special character (e.g., "· · · "), and the other is the pattern of reusing substrings (e.g., "+−+−").
2) *Continuous Keyboard Input (CKI) Sequence:* We considered situations where special characters are adjacent on the U.S. keyboard (e.g., "!@#" and "/.,"). The adjacency here only refers to the left and right directions. This is so because we find that the efficiency of the semantic extraction algorithm in matching strings of CKI sequences decreases drastically after considering the up and down directions to be adjacent, and the number of up and down adjacent CKI sequences that are successfully matched is extremely small. In addition, the total number of CKI sequences adjacent to each other in the right-to-left direction is less than 10% of the left-to-right direction.
3) *Meaningful Sequence:* In this case, we created a list of patterns using 35 of the most common and meaningful special strings gathered from the Internet and the training set. The aim was to match as many meaningful special character segments as possible in the data set. The match list comprises four primary categories of significant strings: strings with mathematical implications (e.g., "+−*/"), strings with left-right symmetry (e.g., "[()]"), strings with address-specific format (e.g., "://"), and strings with facial expression-like patterns (e.g., "@_@").
4) *Multitag Sequence:* Besides appearing alone in passwords, the above three tags may also be combined arbitrarily (e.g., the combination of ".." and "!@" in "Love520..!@").

2) Semantic Characteristics Extraction Algorithm: While there exist many special segments of length 1 in the passwords, they cannot form a semantic pattern by themselves. Moreover, some special segments of length 5 or over 6 may possess some semantic patterns, but due to their extremely low frequency in data sets, we did not incorporate them while developing our algorithm.

First, considering the repeating patterns in special segments (e.g., "!@#!@#"), we treat "!@#" as the special segment that actually determines the semantic type. Therefore, we would first use the Knuth–Morris–Pratt (KMP) algorithm [44] to extract the loop section in the special segment. This will greatly reduce the repetitive operations generated by Algorithm 1 during the process of matching strings, and substantially improve the matching efficiency of Algorithm 1. It should be noted that the four semantic tags would have overlapping areas. For instance, "()" is not only a Meaningful Sequence, but also a CKI special sequence. To this end, when we designed the module for detecting semantic tags (the Detect() in Algorithm 1), we choose to dynamically adjust the order of detecting semantic tags by changing the priority parameters, and return the result immediately when a tag is matched. In the Detect(speSeg, Prio), we provide three matching priorities: Default, Length$_4$ and Length$_6$. When the

Algorithm 1: Special Segment Semantic Extraction

```
Input: A preprocessed dataset, $S$
Output: Semantic tags lists and Semantic string lists of
        the special segment from $S$, $pwdTagList$,
        $SemanticList$
1  $pwdTags$ ←all tags of a password;
2  $orders$ ← ImportOrders();
3  $Prios$ ←Priority of matching tags, defaults to "Default";
4  for all password such that $password \in S$ do
5      for each special segment specialSegment in password do
6          if $len(specialSegment) < 2$ or $> 6$ or $== 5$ then
7              continue
8          $semanticTag$ ←NULL;
9          $speSeg$ ← KMP($specialSegment$);
10         if $len(speSeg) \neq len(specialSegment)$ then
11             $semanticTag$ ←"Reque Tag";
12             continue
13         if $len(speSeg)$ == 4 or 6 then
14             $Prios$ ← $Length_4$ or $Length_6$;
15         $semanticTag$ ← Detect($speSeg, Prios$);
16         $nowOrder$ ← SetOrder($speSeg, orders$);
17         for each matching function func in nowOrder do
18             $result$ ← $func(speSeg)$;
19             if $result_{[0]}$ is true then
20                 $semanticTag$ ← $result_{[1]}$;
21                 $SemanticList_{[len(speSeg)]}$.insert($speSeg$);
22             $pwdTags_{[semanticTag]}$++;
23     $pwdTagList_{[len(speSeg)]}$.append({'pwd': $password$,
       'pwdTags': $pwdTags$});
24 return $pwdTagList$, $SemanticList$
```

segment length is 2 or 3 (priority is the Default), we prioritize the matching Reuse Sequence, followed by the CKI Sequence. When the segment length is 4 (priority is the $Length_4$), we prioritize the matching Meaningful Sequence and the CKI Sequence, followed by the Reuse Sequence. When segment length is 6 (priority is the $Length_6$), we prioritize the matching Reuse Sequence and the CKI Sequence, followed by the Multitag Sequence.

By collecting meaningful special string data on the Internet and password data sets (CSDN, Youku, Myspace, and Zoosk), we constructed a dictionary with 100 special strings with special meanings, which is used to match Meaningful Sequences within the special segments. Easily, we can combine length ranges and regular expressions (refers to the necessary rules for algorithm matching and Reuse Sequence patterns, including the rules consisting of only a single character element and the rules that the character element must be a popular special character.) to match single Reuse Sequence-type special segments. For the matching of CKI Sequences, we need to consider the position of the first special. Because the distribution of specials in computer keyboards is not completely centralized, half of them are distributed on the second line of the keyboard, and half are distributed at the end of lines 3–5 of the keyboard. When matching a Multitag Sequence, it may be broken down

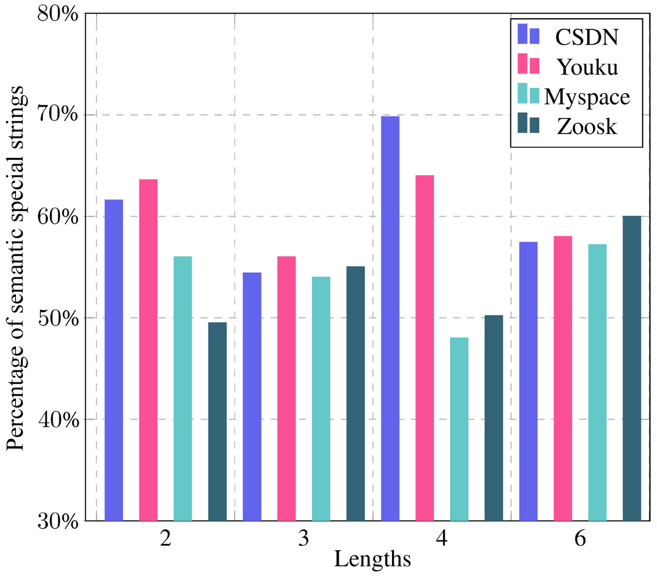

Fig. 2. Proportion of strings with semantic patterns in popular length segments.

into a combination of different subtags. At this point, we will use the methodology proposed by Li et al. [41] to calculate the semantic coverage of strings to select the combination with the highest semantic coverage (e.g., "!@.." has two semantic words "!@" and "..". According to Li's semantic coverage formula, the coverage rate of "!@.." is the sum of 2^2 and 2^2 divided by 4^2). The details of the whole extraction process are shown in Algorithm 1.

Since there are likely to be multiple special segments in a password, we use the SetOrder() function to update the new index position of the detection. This facilitates successive matching of the same password by the matching loop matching module (lines 16–21). The algorithm takes the data detected by each loop and stores them in PwdTagLists() and semanticList(), respectively. Where the semantic strings and their corresponding tags are stored in PwdTagList() to better analyze the distribution characteristics of the semantic patterns in different popular length segments. SemanticList() saves all the potential semantic strings in each popular length segment, which are then utilized as the building materials for constructing semantic dictionaries of special segments in the following stage.

3) Semantic Characteristics Analysis: In Fig. 2, the proportion of passwords with special character semantic tags is displayed across various data sets. It is observed that special semantic patterns are detected in only a range of 21.5%–34.7% of the data set. However, detected semantic strings account for 48%–69% of the corresponding length segments, based on the data set and segment length considered. This indicates the validity of restricting the identification of semantic patterns to special character segments of lengths 2, 3, 4, and 6. This not only decreases the quantity of inaccurate detections, thereby enhancing efficiency, but also demonstrates how real users' behavioral patterns in regards to semantic strings vary based on their length. As depicted in Fig. 2, irrespective of the data set, the proportion of semantic strings

TABLE III
DISTRIBUTION OF POPULAR SEMANTIC PATTERNS IN EACH POPULAR LENGTH SEGMENT

Length	Dataset	Semsntic distribution of special strings			
		Reuse	CKI	Meaningful	Multi-tags
2	CSDN	**49.55%**	8.66%	3.69%	
	Youku	**50.64%**	9.03%	4.02%	
	Myspace	**51.67%**	1.89%	2.61%	
	Zoosk	**46.21%**	1.67%	1.92%	
3	CSDN	33.23%	18.07%	3.65%	0.85%
	Youku	36.50%	15.80%	2.93%	0.72%
	Myspace	38.92%	4.72%	1.07%	**10.56%**
	Zoosk	36.73%	4.57%	2.16%	**12.03%**
4	CSDN	27.92%	25.96%	**7.11%**	8.84%
	Youku	28.56%	19.67%	**6.89%**	9.07%
	Myspace	28.76%	4.91%	4.31%	**10.15%**
	Zoosk	29.03%	6.59%	3.31%	**11.27%**
6	CSDN	27.90%	17.32%	0.43%	11.78%
	Youku	30.11%	16.93%	0.87%	10.69%
	Myspace	**40.78%**	9.62%	0.27%	**6.77%**
	Zoosk	**43.97%**	10.92%	0.19%	**5.94%**

* Note: all special segments here have a minimum length of 2.

in segments of length 4 and 6 is greater than that in segments of length 3. This trend suggests that real users are more prone to utilizing semantic patterns as the segment length increases. The study indicates that the CSDN and Youku data sets have a greater proportion of semantic strings in segments of length 4 than other lengths, which contrasts with the Myspace and Zoosk data sets. Furthermore, an overall analysis reveals that the CSDN and Youku data sets possess a higher proportion of semantic strings in popular length segments compared to the Myspace and Zoosk data sets. Both phenomena demonstrate the potential impact of Linguistic background on user behavior when selecting special character semantic strings.

Table III illustrates the distribution of semantic patterns across popular length segments in the data set. It is evident that the Reuse pattern is the most prevalent semantic pattern across all data sets and all popular length segments. This suggests that users generally recognize the creation of special strings through reuse. Additionally, the occurrence of patterns other than Reuse is closely linked to segment length or data set. It is evident that the most prevalent semantic pattern across the various data sets and length segments is Reuse.

1) The occurrence of CKI strings is more common in the Chinese data set than the English data set. The CKI pattern is primarily concentrated in segment of length 3, 4 or 6, particularly in the segment of length 4. However, its occurrence proportion in the English data set is lower.
2) Conversely, the English data set has a higher likelihood of containing Multi-tags strings than the Chinese data set.
3) Meaningful patterns are more likely to occur in segments of length 4.
4) The proportion of CKI patterns in the Chinese data set comes second only to Reuse patterns in the segments of length 3 or 4, while the proportion of Multi-tags patterns comes second only to Reuse patterns in the English data set.

TABLE IV
TOP-3 SEMANTIC STRINGS IN EACH SEMANTIC PATTERN

Dataset	Top-3 semantic strings in each semantic patterns			
	Reuse	CKI	Meaningful	Multi-tags
CSDN	..	!@#	+-*/	.-+
	**	!@	://	..+
	...	!@#$	/*-	..//
Youku	..	!@#	://	++**
	...	!@	+-*/	..//
	!!	!@#$%ˆ	/*+-	.-+
Myspace	!!	!@#	/*-+	\\’
	...	!@#$%ˆ	/*-	**_
	**	!@#$	(*)	!!@@
Zoosk	!!	!@#$%ˆ	-_-	\\’
	**	!@#$	ˆ_ˆ	;\\’
	...	!@#	/*-+	@$$

5) The proportion of Multi-tags patterns in the Chinese data set increases as the segment length increases. This may reflect the tendency of Chinese users to utilize special semantic strings.

Next, we present some of the most commonly detected semantic strings in Table IV. Regardless of the data set, certain semantic strings like “!!!”, “**”, “. . .”, “!@#”, “!@#$”, and “!@#$%^” frequently appear at the top list, indicating they are most likely the most commonly used semantic strings in most cases. Widely recognized special character sequences, such as “+−*/” and “/*−+”, have been proven to be commonly used meaningful strings. Furthermore, our findings reveal that the top semantic strings in the Chinese data set invariably incorporate “://”, whereas such strings are absent in the English data set. Therefore, we speculate that Chinese users are more likely to create passwords using URL strings of Web pages than English users. More interestingly, we found the special strings shaped like a smiley face in Zoosk’s top semantic strings. We believe that this is likely to be influenced by the type of service provided by Zoosk, which is a large social dating app.

IV. PASSWORD ATTACK

A. State-of-the-Art Method

Weir et al. [37] proposed an optimized dictionary guessing method derived from the CFG method for PCFGs, which has been widely used in password research [13], [40]. According to extensive experiments conducted in [45], PCFGs is one of the most efficient password attack methods in terms of the speed of generating correct guesses, which makes it suitable as a basic method for illustrating characteristics attacks. On the other hand, while methods like neural networks can automatically guess from scratch, they learn structures that are not interpretable, which is not conducive to the study of special segment characteristics. We use PCFG$_{v4}$ as a base guess generator to demonstrate scenarios for the use of special characters in attack sequences.

PCFG$_{v4}$ is the latest version of PCFGs proposed by [42] in 2019, which provides a better cracking rate than PCFGs. PCFG$_{v4}$ adds a feature that the passwords are not only grouped as the basic structures of L (letters), D (digits), and S

TABLE V
INSTANCES OF PCFG$_{v4}$ PASSWORD STRUCTURES

Structure	Example
Simple	LYKS
Base	$L_4Y_1K_3S_2$
Pre-terminal	$L_4$2007qaz..
Terminal	Jame2007qaz..

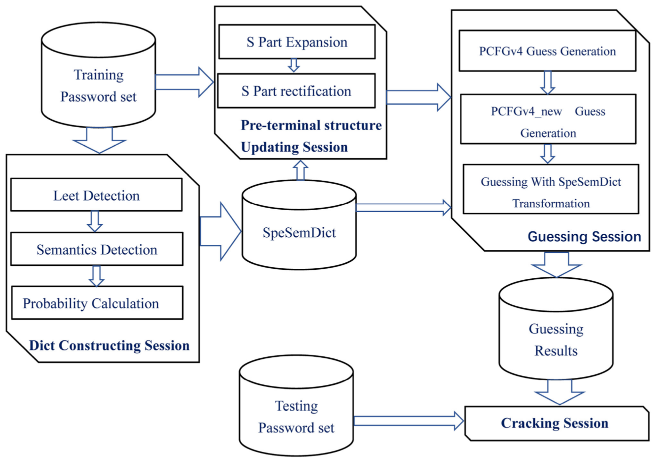

Fig. 3. Flow diagram of guessing and cracking phase in the characteristic attack method of special segments.

(special) in PCFGs, but also with richer structures [including Y (years), K (keyboard), and E (emails)]. Besides, the PCFG$_{v4}$ distinguishes between uppercase and lowercase letters. The PCFG$_{v4}$ method can be divided into three phases: 1) training phase; 2) generation phase; and 3) guessing phase. In the training phase, the PCFG$_{v4}$ method generates lists of different character segments based on the training data, including D-list, S-list, L-list, Y-list, K-list, E-list, and structures list (including simple and basic structures, as shown in Table V). In the generation phase, the PCFG$_{v4}$ method generates the candidate passwords based on the syntax according to the calculated probabilities in descending order. The probability of the candidate password is the product of the probabilities of the corresponding lists. It is worth noting that in the process of generating the candidate passwords, PCFG$_{v4}$ first generates the preterminal structure (the string after all segments except the L-segment have been filled, as shown in Table V) based on the base structure, and then generates the terminal structure (the candidate password generated after the L-segment has been filled) based on the preterminal structure. This is to reduce the impact of the complexity of the L-segments on the efficiency of generating candidate passwords. Finally, the PCFG$_{v4}$ method compares these candidate guessed passwords with the test set passwords to calculate the cracking rate.

B. Our Proposed Method

Our characteristics attack method has four sessions as shown in Fig. 3. We randomly select half of the passwords in each data set as the training set to construct the semantic dictionary and the basic PCFG$_{v4}$ model. The dictionary construction session is capable of outputting semantic dictionaries of special segments. In the session of updating preterminal structure, we extend and correct the S-list of PCFG$_{v4}$ to generate more guessed passwords with special character semantic strings. In the guessing session, we let newPCFG$_{v4}$ guess the training set and then generate the final guesses using the *speSemDict* dictionary (the special segment semantic dictionary). In the cracking session, we attack the test set. In the remainder of this section, we will describe how to 1) construct the dictionary; 2) improve PCFG$_{v4}$ by updating its preterminal structure; and 3) generate new guesses using semantic string transformation.

1) Constructing Special Segment Semantic Dictionary: To further investigate the factors influencing the strength of passwords with special characters, we constructed a semantic dictionary of special segment based on their semantic characteristics. The construction method considers not only the Leet patterns (which refers to the character transformation patterns motivated by glyph similarity or other similarities [14]) employing special characters, but also all frequently occurring semantic patterns of special segment. It enables the creation of a semantic dictionary for a specific data set. It is divided into three primary phases. The first phase identifies the passwords with special character Leet patterns in the data set and calculates the probability of encountering these Leet special characters. The second phase identifies all the semantic strings in the data set that meet the semantic pattern and determines their frequency of occurrence. In the third stage, the semantic probability coverage formula is applied to compute the probability of using each element in the list of popular length segments. Subsequently, *speSemDict* dictionary is generated.

2) Updating the Preterminal Structure of PCFG$_{v4}$: In order to improve the cracking rate of PCFG$_{v4}$ for passwords with special characters, we have targeted extensions and corrections to the S-list the original PCFG$_{v4}$.

First, we extracted the data that is available in *semanticList* (derived from the output of Algorithm 1) and the S-list data obtained by PCFG$_{v4}$ training (*Slist*), and then inserted these two parts of the data into a new list (*Newlist*). Subsequently, we update the frequency of occurrence of all strings in the *NewSlist* by naming the updated *Newlist* as *NewSlist*. Immediately after that, we populate the *NewSlist*$_{[0]}$ section with the *speSemDict*$_{[1]}$ obtained from the training set to increase the probability of using leet characters (lines 16–23 in Algorithm 2). Finally, the preterminal structure of PCFG$_{v4}$ is updated in conjunction with the new *NewSlist*. It should be emphasized that the prerequisite for updating the PCFG$_{v4}$ preterminal structure using this method is that the training set used for constructing the dictionary must be consistent with the training set used for PCFG$_{v4}$. It is also worth stating that we have only added the S-list extension and correction module to the original PCFG$_{v4}$ algorithm for generating new preterminal structure, with no increase in time complexity compared to the original algorithm and no substantial change in the size of the objects that the algorithm handles. As we can see from Table VI, the S-list of each data set is not expanded much, on average, nearly 280 per data set, and the expansion is

TABLE VI
CHANGES CAUSED TO THE ORIGINAL PCFG$_{v4}$ GRAMMARS

Training set		CSDN	Youku	Myspace	Zoosk
Base structures		21,201+0	23,054+0	45,391+0	65,291+0
L-lists		230,489+0	251,037+0	451,037+0	585,473+0
D-lists		876,633+0	934,772+0	391,477+0	463,261+0
Y-lists		71+0	67+0	194+0	207+0
K-lists		712+0	698+0	914+0	954+0
E-lists		341+0	397+0	107+0	123+0
S segments	Length2	551+17	603+22	837+41	678+57
	Length3	732+43	719+56	1,927+101	1,328+87
	Length4	599+67	632+83	1751+159	1573+136
	Length6	164+39	161+52	267+96	193+74
	All	2907+166	3146+213	6,168+397	5,076+354

Algorithm 2: Improved PCFGsv4 Attack

```
   Input: $speSemDict$, $semanticList$, $\tau$, $Guessset$;
   Output: A set of possible guess passwords, $PWD$; A
           new guess set, $NewguessSet$;
1  Generate possible guess passwords($p_0$)
2  $replaceSet \leftarrow Guessset_{[0:\tau\times\beta]}$;
3  for all pwd such that $pwd \in replaceSet$ do
4      $Orders, Lengths, Numbers, Locations \leftarrow$
       findSpeSegment($pwd$);
5      if $\frac{Lengths}{len(pwd)} < \mu$ and $Numbers \neq NULL$ and
       $1 < Lengths < 7$ then
6          if $Numbers == 1$ and $Locations \neq$ "Head" then
7              $Strs_1 \leftarrow speSemDict_{[Lengths][0]}$.get('Strs');
8              $p_0 \leftarrow$
               speSegmentByStrs($pwd, Locations, Strs_1$);
9              $PWD$.Add($p_0$);
10         if $Numbers > 1$ and $Locations ==$ "Head-Tail
           symmetry" then
11             $Strs_1 \leftarrow speSemDict_{[Lengths_{[0]}][0]}$.get('Strs');
12             $Strs_2 \leftarrow speSemDict_{[Lengths_{[1]}][0]}$.get('Strs');
13             $p_0 \leftarrow$
               spesegmentByStrs($pwd, Locations, Strs_1, Strs_2$);
               PWD.Add($p_0$)
14         return $PWD$
15 $NewguessSet \leftarrow$ transGuessData($Guessset, PWD, \tau, \beta$);
16 return $NewguessSet$
17 Update the pre-terminal structure of PCFG$_{v4}$:
18 for $Strs \in semanticList$ do
19     if $Strs \notin Slist$ then
20         Newlist$_{[len(Strs)]}$.insert($Strs$);
21 for $Strs \in Newlist$ do
22     $f(Strs) \leftarrow \frac{total(Strs)}{total(Ssegment_{[len(Strs)]})}$;
23     NewSlist$_{[len(Strs)]}$.insert({'Str':$Strs$, 'Pro':f($Strs$)});
24 NewSlist$_{[1]} \leftarrow$ spesemDict$_{[1]}$;
```

concentrated in the S-list of length 2, 3, 4, and 6. Therefore, we believe that this method is a highly feasible one, which not only can theoretically generate more guessed passwords with special character semantics quickly, but also does not incur too much extra processing load. In this article, in order to better distinguish PCFG$_{v4}$ attacks that use different preterminal structures, we refer to the attack that uses *NewSlist* as the NewPCFG$_{v4}$ attack.

3) Transformation Guessing Attack Based on Semantic String Transformation: We feed the output of PCFG$_{v4}$ cracker (*Guessset*) and the output of construct semantic dictionary algorithm (*speSemDict*) to this phase as input. Assume we will attack τ times, and the replacement rate is β [40]. Namely, we select the top β percent of the τ passwords, which are more likely to appear in the real password set, as the raw material to perform semantic transformation. Then, we use the passwords after semantic transformation to substitute the same number of guessing results with the overall lowest possibility in PCFG$_{v4}$, which is the last β percent of the τ passwords. It is noted that the basic structure of the transformed new passwords and the data in the D, L, E, K, and Y segments are identical to those of the pretransformation passwords. We refer to this attack method as speSemDict-PCFG$_{v4}$.

We show how to perform semantic transformation based on particular password in Algorithm 2. The high-level idea is that we detect objects in the password that meet the transformation conditions, then alter them to generalize guesses.

Algorithm 2 uses *speSemDict* and *Guessset* as inputs. *Guessset* can be the result of guesses from other tools. First, we check whether the length of the special segment satisfies the transformation rule. If the segment length is not between 2, 3, 4, and 6, we skip this password. This is for algorithmic time efficiency reasons. In addition, if the ratio of the segment length to the total password length reaches the exit ratio μ, we also skip this password. This is because when the ratio reaches μ, the algorithm will produce a large number of invalid transformations, which will directly affect the accuracy of the algorithm. It is noted that μ is a variable parameter, and the exact value should be flexible to the data set. By conducting experiments on the four data sets in Section III-A, we found a law that prevails for the μ value. That is, as the μ value gradually increases from 0.1, the attack accuracy of the algorithm first increases and then decreases, reaching a maximum between 0.3 and 0.4. To facilitate the next attack experiments, we took the average of the best μ values corresponding to the four data sets, 0.33, as the common parameter value of μ in this study.

Then, we try to control the transformation behavior of the semantic string using the *Numbers* and *Locations* generated by findSpeSegment(). Assuming that *Numbers* is 1 and *Locations* is either "End" or "Middle," or that *Number* is 2 and *Locations* is "Head-Tail symmetry," specSegmentByStrs() is triggered to complete the transformation of special segments. The purpose of limiting the range of transformation operations is to balance the conflict between algorithmic accuracy and time efficiency. In specSegmentByStrs() we replace the special character segments in the original *pwd* with the semantic strings (Str_i) with the highest probability in the corresponding sublist of the *specSemDict* to generate new passwords p_0 and add each generated p_0 to *PWD* (lines 6–15). In the transGuessData() function, we store the top τ passwords extracted from Guessset in the replacement-list, then replace the last $\tau \times \beta$ data in the

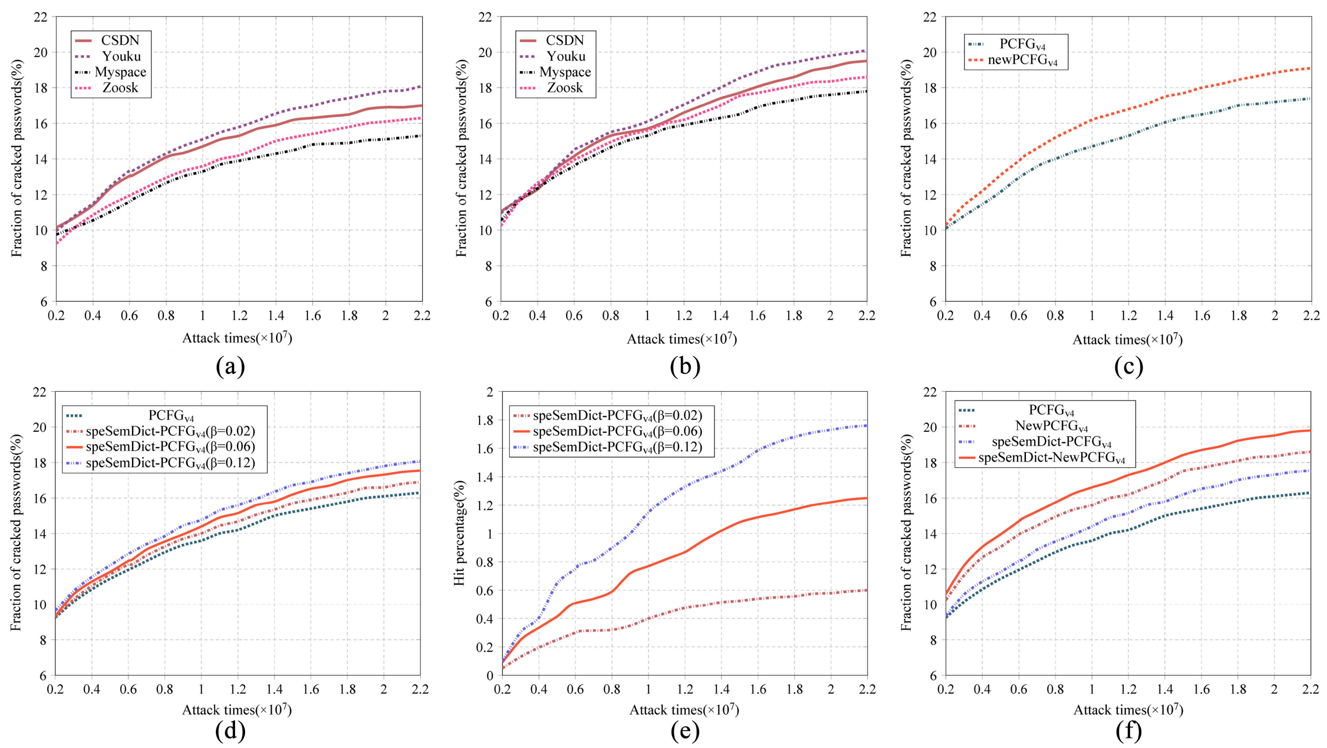

Fig. 4. Cracking results. (a) and (b) Cracking results of basic PCFG$_{v4}$ and our improved NewPCFG$_{v4}$ on different data sets. (c) Tangible advantages obtained by the NewPCFG$_{v4}$ attacks. (d) PCFG$_{v4}$ versus transformation attacks (i.e., speSemDict-PCFG$_{v4}$ attacks) with different β. (e) Hit percentages of speSemDict-PCFG$_{v4}$ with different β. (f) Cracking results of all improved PCFG$_{v4}$ attacks ($\beta = 0.06$).

replacement list with the data in *PWD*, and finally pass all the data in the replacement-list to the *Newguessset*, and empty the replacement-list.

V. Experiments and Results

A. Experimental Data Set

We randomly divide all four data sets mentioned in Section III-A into two equal parts. One is used as a training set and the other is used as a test set. The training and test sets are mutually exclusive. Note that the training set below is exactly the training set mentioned in the previous section.

1) *CSDN:* 208 609 passwords with specials, including 104 304 as training set.
2) *Youku:* 2 870 066 passwords with specials, including 1 435 033 as training set.
3) *Myspace:* 13 128 817 passwords with specials, including 6 564 408 as training set.
4) *Zoosk:* 3 274 877 passwords with specials, including 1 637 438 as training set.

The main objective of this experiment is to explore the actual attack effect of the attack method incorporating special segment characteristics on passwords with special characters, so all the data we choose for the training set should contain special character segments. This choice of training set not only better reveals the actual attack effect of our proposed attack method, but also improves the efficiency of the attack experiment by reducing the number of irrelevant guesses. We choose to use the original PCFG$_{v4}$ attack as a basic control experiment, and derive the specific effectiveness of the new attack methods through the comparison of experimental results. The use of control experiments can effectively reduce the influence of other factors (e.g., the characteristics of L, D, Y, K, and E segments) on the effectiveness of this experiment.

B. Cracking Results

1) NewPCFG$_{v4}$ Attack Cracking Results: In Fig. 4(a)–(c), we compare the performance of the original PCFG$_{v4}$ and the PCFG$_{v4}$ with the updated preterminal structure (NewPCFG$_{v4}$) using the Zoosk training set as an example, since it is the largest training set considered in this article. Fig. 4(a) shows the cracking results of the original PCFG$_{v4}$ in the offline attack. The cracking rate increases quickly because they always try high probability guesses first. Fig. 4(b) shows that NewPCFG$_{v4}$ exhibits little improvement when the number of allowed guesses is small (e.g., below 4×10^6); the improvement increases as the number of guesses increases. For example, with 10^7 guesses, the success rate improves by 0.65%–1.24%; with 2.2×10^7 guesses, this number reaches 2.07%–3.01%. This suggests that the use of frequently used special character characteristics helps to reduce the searching space for special character segments, which is particularly important when a large number of guesses is allowed. Moreover, our improved attack also demonstrates its advantages despite the fact that no suitable training set is available [see Fig. 4(c)].

2) Transformation Guessing Attacks Cracking Results: In Fig. 4(d)–(f), we compare the performance of the original PCFG$_{v4}$ and the PCFG$_{v4}$ with transformation guessing (speSemDict-PCFG$_{v4}$ or speSemDict-NewPCFG$_{v4}$) using the Zoosk training set and test set as examples. We examine the impact of the transformation rate, β, on cracking. Three β values are tested: 0.02, 0.06, and 0.12. We calculated the associated cracking rate for each β with varying numbers of guesses. Originally, both methods produced similar results. This can be attributed to the small value of $\tau \times \beta$, regardless of the value of β. That is, the number of high-probability passwords that can be used as transformation material is not enough for the speSemDict-PCFG$_{v4}$ attack method to generate more guessing passwords that can successfully match the test set. As the cracking process continues, the generation probability of guesses diminishes, leading to a slower rate of increase. Fig. 4(d) depicts that approximately 10^7 attacks later, transforming 6% of low probability guessed passwords transformation facilitates the cracking of 0.55% more real passwords in the test set.

To investigate the effectiveness of the newly generated passwords by speSemDict-PCFG$_{v4}$ in attacks, the hit rate is defined as the ratio of the total number of successful matches with the test set to the total number of passwords in the test set. The generation rate fluctuates based on the number of attacks conducted in various β experiments, and the findings are illustrated in Fig. 4(e). The generation rate rises as β becomes larger. Nonetheless, the efficacy of transformation decreases with β. Therefore, we set β to 0.06 in subsequent experiments to balance transformation efficiency and matching accuracy.

Fig. 4(f) illustrates the efficacy of three enhanced PCFG$_{v4}$ attacks. The speSemDict-NewPCFG$_{v4}$ attack denotes the use of the Guessset generated by NewPCFG$_{v4}$ as the data source for the transformation guessing attack. Two noteworthy facts are observed here. First, the original PCFG$_{v4}$ requires more guesses than any other improved attack to attain a comparable success rate. It has been demonstrated that adding special segment characteristics to the existing attack model can improve the success rate of cracking passwords with special characters. Moreover, the speSemDict-NewPCFG$_{v4}$ attack surpasses the effectiveness of both the speSemDict-PCFG$_{v4}$ and NewPCFG$_{v4}$ attacks when it comes to cracking. It appears that the updated preterminal structure method and improved special character segment transformation guessing method can be combined to maximize the success rate of cracking password with special characters. However, our improved attack method for PCFG$_{v4}$ is less portable. We suspect that this is due to the fact that the *speSemDict* used in the new attack method is more highly correlated with a specific set of users, and therefore more difficult to port.

VI. Discussion

A. Password Creation Policy Survey

1) Background of the Survey: Password creation strategies have always played a great role in effectively avoiding weak password attacks, and this is no exception in IoT platforms and smart contract wallets. The password creation policies of IoT platforms and smart contract wallets can guide users to create passwords that are not easy to attack, which safeguards their data and reduces the risk of their data being leaked, thus increasing their trust in the platform. However, Our experimental results in the previous section also confirm that, with the continuous exploitation of more effective password characteristics and the optimization of advanced auto-learning attack algorithms, the previous popular password creation policy standards are gradually insufficient to guarantee the true strength of user passwords. Therefore, continually evaluating and improving current password creation policies is important for the development of IoT and smart contract platforms as well as for the increase of user trust in the platforms.

2) Method of the Survey: In the previous section, our experiments confirmed that passwords with common characteristics of special character segments are insecure. In order to verify the reasonableness of the current password creation policies of IoT platforms and smart contract wallets in guiding users to create special character segments, we conducted a special survey on the current password creation policies of IoT platforms and smart contract wallets. The survey includes feedback on the strength of passwords, and the main rules for restricting special character segments.

First, we generate four guessing sets corresponding to the four training sets using the PCFG$_{v4}$ attack, then select Top-10 guessing passwords from each guessing set, and finally delete the duplicates to generate 35 unique vulnerable passwords with special characters. Second, we utilize a batch registration tool to register 35 accounts containing vulnerable passwords in each IoT platform or smart contract wallet in turn, and generalize the feedback received during the registration process. We call vulnerable passwords with strength feedback results of "average," "medium," "high," and "above 60" as overestimated passwords. Finally, we present the feedback results related to special character segments in Table VII. where the level of overestimation is divided based on the number of overestimated passwords as a percentage of the total number of vulnerable passwords.

3) Results of the Survey: From Table VII, we can clearly see that the password creation policies of some IoT platforms and smart contract wallets have restricted a few insecure special field creation patterns (e.g., reuse patterns), but it is worth noting that restricting only these few special segment characteristics listed in the table is not enough to guarantee the strength of passwords with special characters. As shown in Table VII, the password creation policies of these IoT platforms and smart contract wallets overestimate, to varying degrees, the actual strength of vulnerable passwords with special characters. This risk, if exploited by hackers, will not only affect users' trust in the platform, but also pose a threat to the security of the IoT and smart contract platform.

Our survey results show that the current password creation policies of most IoT platforms and smart contract wallets still have a lot of deficiencies in guiding users to create special character segments. In our future work, we will propose concrete and feasible solutions based on the results of our current research on how to improve the current password creation policies. We also urge IoT platform administrators to pay attention to the above risks as soon as possible.

TABLE VII
PASSWORD STRENGTH OVERESTIMATION LEVEL IN POPULAR IoT PLATFORMS AND SMART CONTRACT WALLETS

Platforms & Wallets	Rules for detecting Symbols	Overestimation level*
AWS IoT Core	Symbols; Middle symbols; Sequential Symbols(3+);	**Middle**
Azure IoT	Symbols; Dictionary or Brute-force attack check;	**Low**
Huaweicloud IoTDA	Symbols; Symbols repeated;	**Middle**
Tencent IoT Explorer	Symbols; Poor practicality; Check leaked passwords;	**High**
Aliyun IoT	Symbols; Symbols repeated;	**Low**
IBM Watson IoT	Symbols; Check common passwords;	**Middle**
Coinbase Wallet	Symbols; Symbols repeated;	**Middle**
MetaMask Wallet	Symbols; Brute-force attack check;	**Low**
Trust Wallet	Symbols; Sequential Symbols(3+);	**Middle**

* "High" refers to the number of vulnerable passwords with overestimated password strength exceeded 66.6% of the total number of vulnerable passwords, and "medium" refers to the number of vulnerable passwords whose strength is overestimated accounts for 33.3% ~ 66.6% of the total number of vulnerable passwords.

TABLE VIII
RESULTS OF PASSWORD CRACKING FOR DIFFERENT BLACKLIST SIZES

Cross attack	test set*	blacklist sizes				
		0	10	30	50	100
Cracking rate	CSDN with dup-data	19.47%	18.95%	18.49%	18.03%	18.16%
	CSDN without dup-data	18.16%	17.82%	17.46%	17.05%	17.35%
	Youku with dup-data	20.39%	19.83%	19.50%	19.07%	18.49%
	Youku without dup-data	19.01%	18.76%	18.51%	18.12%	17.68%

* The test set with dup-data refers to the test set contains identical passwords.

B. Impact of Blacklists on Password Security

As mentioned in [26], the use of blacklists can achieve a balance between password usability and security. Therefore, inspired by the approaches in [26], we used the generic *speSemDict* mentioned in the previous section to construct special character blacklists to better help IoT platforms to avoid the above-mentioned risks. We extracted the most common 10, 30, 50, and 100 special character semantic strings in the generic *speSemDict* as blacklists of different sizes. We used the CSDN and Youku data sets to analyze the effect of different sizes of blacklists on the security of passwords with special characters. The results, as shown in Table VIII, show that the cracking rate of passwords in the test set with duplicate data and the test set without duplicate data decreases gradually as the size of blacklists increases. Therefore, setting up a special character blacklist decreases the success rate of cracking passwords with special characters.

C. Scalability

In Fig. 3, it can be observed that the "Dict Construction" session, the "Update Preterminal Structure" session, and the "Guessing" session are detached from the "Attacking" session. This implies that only the "Attacking" session concerns the attack method. As a result, the output from the "Guessing" session can also be utilized by well-known methods like JtR [34] and neural networks. Furthermore, the *speSemDict* remains unconnected to the underlying method. It can enhance algorithms that ignore common special character characteristics.

D. Password Protection

Our research aims to enhance the comprehension of the utilization of special character segments (both basic and semantically common) in passwords. This development is expected to benefit system administrators and security researchers working on IoT control systems, leading to the implementation of more effective measures for strengthening passwords. Currently, the basic attack dictionary is inadequate, and a generic *speSemDict* would be helpful to adversaries. On the other hand, since neural network password attacks or traditional password characteristic attacks preferentially generate high-probability guessing passwords by means of learning high-frequency password characteristics in the data set, the user's method of using some infrequently used semantic patterns equipped with some lesser-known semantic strings of special characters can effectively enhance the strength of passwords with the limited computing power of the current computers (i.e., the order of magnitude of the attack guess is less than 10^9 [22]). In addition, since the semantic strings themselves have a fixed syntactic format, users can quickly create and change their personal passwords by simply memorizing the fixed syntactic format.

Our research shows that studying the underlying characteristics and semantic patterns of special character segments is a double-edged sword in password security. First, these can be easily integrated into existing password cracking methods (e.g., PCFGs) to enhance long-term efficiency. On the contrary, infrequent special character segment characteristics could assist users in reinforcing their passwords, frequent special character segment characteristics could assist IoT platforms to generate blacklists with such elements.

E. Limitations

Our study did not take into account the impact of users' age, gender, and occupation on their selection of special segments, which could result in an overestimation of the issue. However, to increase efficiency and effectiveness, our research exclusively concentrates on the common semantic characteristics of popular length segments that occur with greater frequency. On the other hand, it is possible that segments of other lengths may exhibit particular semantic patterns that could result in underestimation issues.

F. Ethical Considerations

The use of leaked data sets has become a common practice in password research. Nonetheless, we recognize the ethical issues involved in studying leaked passwords. To address these concerns, each data set in our study has been stored and utilized diligently, only for research purposes. Our study does not reveal any passwords or utilize the information in any way beyond the confines of research.

VII. Conclusion

In this article, we have presented a comprehensive study of special character segments in passwords. The study represents the most extensive and orderly analysis of the security of special character segments to date, and is systematic and thorough. Based on the results of the password analysis, we found that users of different languages tend to use different special character segments when creating passwords. It has been shown that the inclusion of special character segments in the password attack model can significantly increase the success rate of password cracking. This improvement depends largely on the variability of the data set and semantic dictionary used.

Our survey found that current password creation policies for mainstream IoT platforms and smart contract wallets somewhat overestimate the strength of weak passwords with special characters. We recommend IoT and smart contract platforms administrators to pay attention to this risk as soon as possible. The study experimentally confirms the effectiveness of using special character blacklists to decrease the vulnerability of passwords featuring special characters being cracked. We strongly recommend IoT and smart contract platforms to incorporate efficient special character blacklists to restrict the utilization of the vulnerable special character strings disclosed in this article. Moreover, Internet users should refrain from using insecure password policies made public in this study while creating or updating their login passwords.

References

[1] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A survey of Internet of Things (IoT) authentication schemes," *Sensors*, vol. 19, p. 1141, Jan. 2019.

[2] A. M. Antonopoulos, and G. Wood, *Mastering Ethereum: Building Smart Contracts and DApps*. Sebastopol, CA, USA: O'reilly Media, Inc., 2018.

[3] P. Praitheeshan, L. Pan, and R. Doss, "Security evaluation of smart contract-based on-chain ethereum wallets," in *Proc. Int. Conf. Netw. Syst. Secur.*, 2020, pp. 22–41.

[4] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jan. 2015.

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

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

[7] S. Gaw and E. W. Felten, "Password management policies for online accounts," in *Proc. 2nd Symp. Usable Priv. Secur.*, 2006, pp. 44–55.

[8] C. Shen, T. Yu, H. Xu, G. Yang, and X. Guan, "User practice in password security: An empirical study of real-life passwords in the wild," *Comput. Secur.*, vol. 61, pp. 130–141, Aug. 2016.

[9] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 2776–2791, 2017.

[10] S. Houshmand, S. Aggarwal, and R. Flood, "Next gen PCFG password cracking," *IEEE Trans. Inf. Forensics Security*, vol. 10, pp. 1776–1791, 2015.

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

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

[13] D. Wang, P. Wang, D. He, and Y. Tian, "Birthday, name and bifacial-security: Understanding passwords of chinese Web users," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 1537–1555.

[14] "Hashcat advanced password recovery." 2019. [Online]. Available: https://hashcat.net/hashcat/

[15] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled Web of password reuse," in *Proc. Netw. Distrib. Syst. Symp.*, vol. 14, 2014, pp. 23–26.

[16] B. Ur, "Supporting password-security decisions with data," M.S. thesis, PNC Center Financ. Services Innov. Microsoft Res., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2016.

[17] C. Wang, S. T. 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. Priv.*, 2018, pp. 196–203.

[18] J. Blocki, B. Harsha, and S. Zhou, "On the economics of offline password cracking," in *Proc. IEEE Symp. Security Privacy*, 2018, pp. 853–871.

[19] 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.*, 2018, pp. 1–6.

[20] B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz, "PassGAN: A deep learning approach for password guessing," in *Proc. 17th Int. Appl. Cryptogr. Netw. Secur.*, 2019, pp. 217–237.

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

[22] H. Shi, W. Zhang, Z. Zhang, and D. Ding, "Vulnerability analysis of chinese digital passwords related to ATM PIN using deep learning," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 4, pp. 2825–2835, Jul./Aug. 2023.

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

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

[25] P. Markert, D. V. Bailey, M. Golla, M. Dürmuth, and A. J. Aviv, "This pin can be easily guessed: Analyzing the security of smartphone unlock pins," in *Proc. IEEE Symp. Security Privacy*, 2020, pp. 286–303.

[26] P. Markert, D. V. Bailey, M. Golla, M. Dürmuth, and A. J. Aviv, "On the security of smartphone unlock pins," *ACM Trans. Priv. Secur.*, vol. 24, no. 4, pp. 1–36, 2021.

[27] M. AlSabah, G. Oligeri, and R. Riley, "Your culture is in your password: An analysis of a demographically-diverse password dataset," *Comput. Secur.*, vol. 77, pp. 427–441, Aug. 2018.

[28] E. Gerlitz, M. Häring, and M. Smith, "Please do not use '!' '?' '_' or your license plate number: Analyzing password policies in German companies," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 17–36.

[29] A. Hanamsagar, S. S. Woo, C. Kanich, and J. Mirkovic, "Leveraging semantic transformation to investigate password habits and their causes," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2018, pp. 1–12.

[30] B. Ur et al., "'I' added '!' at the end to make it secure: Observing password creation in the lab," in *Proc. 11th Symp. Usable Priv. Secur.*, 2015, pp. 123–140.

[31] R. Veras, C. Collins, and J. Thorpe, "On semantic patterns of passwords and their security impact," in *Proc. Netw. Distrib. Syst. Symp.*, 2014, pp. 1–16.

[32] 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. Human Factors Comput. Syst.*, 2016, pp. 3748–3760.

[33] Y. Yang, K. C. Yeo, S. Azam, A. Karim, R. Ahammad, and R. Mahmud, "Empirical study of password strength meter design," in *Proc. 5th Int. Conf. Commun. Electron. Syst.*, 2020, pp. 436–442.

[34] "John the Ripper password cracker." 2013. [Online]. Available: https://www.openwall.com/john/

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

[36] C. Castelluccia, A. Chaabane, M. Dürmuth, and D. Perito, "When privacy meets security: Leveraging personal information for password cracking," 2013, *arXiv:1304.6584*.

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

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

[39] W. Han, M. Xu, J. Zhang, C. Wang, K. Zhang, and X. S. Wang, “TransPCFG: transferring the grammars from short passwords to guess long passwords effectively,” *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 451–465, 2021.
[40] W. Li and J. Zeng, “Leet usage and its effect on password security,” *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2130–2143, 2021.
[41] Y. Li, H. Wang, and K. Sun, “Personal information in passwords and its security implications,” *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 2320–2333, 2017.
[42] “Pretty cool fuzzy guesser.” Github.com. 2019. [Online]. Available: https://github.com/lakiw/pcfg_cracker/
[43] H. Zhang, C. Wang, W. Ruan, J. Zhang, M. Xu, and W. Han, “Digit semantics based optimization for practical password cracking tools,” in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2021, pp. 513–527.
[44] R. Rahim, Z. Iskandar, and J. Hendra, “A review: Search visualization with Knuth Morris Pratt algorithm,” *IOP Conf. Series, Mater. Sci. Eng.*, vol. 237, no. 1, pp. 12–26, 2017.
[45] T. Lundberg, “Comparison of automated password guessing policies,” M.S. thesis, Dept. Electr. Eng., Linköping Univ., Linköping, U.K., 2019.

Daojing He (Member, IEEE) received the B.Eng. and M.Eng. degrees in computer science from Harbin Institute of Technology, Harbin, China, in 2007 and 2009, respectively, and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2012.

He is currently a Professor with the School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen, China. His research interests include network and systems security.

Prof. He is on the editorial board of some international journals, such as IEEE NETWORK.

Zhiyong Liu was born in 1999. He is currently pursuing the master’s degree with the School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou, China.

Shanshan Zhu was born in 1985. She is currently pursuing the Ph.D. degree with Harbin Institute of Technology, Shenzhen, P.R. China.

Sammy Chan (Senior Member, IEEE) received the B.E. and M.Eng.Sc. degrees in electrical engineering from the University of Melbourne, Melbourne, VIC, Australia, in 1988 and 1990, respectively, and the Ph.D. degree in communication engineering from the Royal Melbourne Institute of Technology, Melbourne, in 1995.

Since December 1994, he has been with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong, where he is currently an Associate Professor.

Mohsen Guizani (Fellow, IEEE) received the B.S. (with Distinction) and M.S. degrees in electrical engineering and the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively.

He is currently with the Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE.

Dr. Guizani is a Senior Member of ACM.