Chunk-Level Password Guessing: Towards Modeling Refined Password Composition Representations

Ming Xu, Chuanwang Wang, Jitao Yu, Junjie Zhang, Kai Zhang, Weili Han

Laboratory for Data Analytics and Security, Shanghai Key Laboratory of Data Science, Fudan University

ABSTRACT

Textual password security hinges on the guessing models adopted by attackers, in which a suitable password composition representation is an influential factor. Unfortunately, the conventional models roughly regard a password as a sequence of characters, or natural-language-based words, which are password-irrelevant. Experience shows that passwords exhibit internal and refined patterns, e.g., "4ever, ing or 2015", varying significantly among periods and regions. However, the refined representations and their security impacts could not be automatically understood by state-of-the-art guessing models (e.g., Markov).

In this paper, we regard a password as a composition of several chunks, where a chunk is a sequence of related characters that appear together frequently, to model passwords. Based on the concept, we propose a password-specific segmentation method that can automatically split passwords into several chunks, and then build three *chunk-level* guessing models, adopted from Markov, Probabilistic Context-free Grammar (PCFG) and neural-network-based models. Based on the extensive evaluation with over 250 million passwords, these *chunk-level* models can improve their guessing efficiency by an average of 5.7%, 51.2% and 41.9%, respectively, in an offline guessing scenario, showcasing the power of a suitable password representation during attacks. By analysing these efficient attacks, we find that the presence of common chunks in a password is a stronger indicator for password vulnerability than the character class complexity. To protect users against such attacks, we develop a client-side and real-time password strength meter to estimate the passwords' resistance based on *chunk-level* guessing models.

CCS CONCEPTS

• **Security and privacy**;

KEYWORDS

Password Composition; Chunks; Data Driven Guessing Models

ACM Reference Format:

Ming Xu, Chuanwang Wang, Jitao Yu, Junjie Zhang, Kai Zhang, Weili Han, Laboratory for Data Analytics and Security, Shanghai Key Laboratory of Data Science, Fudan University, . 2021. Chunk-Level Password Guessing: Towards Modeling Refined Password Composition Representations. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3460120.3484743

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8454-4/21/11…$15.00

https://doi.org/10.1145/3460120.3484743

1 INTRODUCTION

Human-created textual passwords are still widely used and will likely remain the dominant authentication mechanism in the foreseeable future for their low deployment costs and remarkable simplicity [13, 14, 41, 63, 68]. Attackers use increasingly advanced tools to guess passwords, with data-driven probabilistic guessing models being the latest development [43, 47, 59]. By modeling the probability distribution behind a set of observed passwords, these models perform efficient guessing attacks by generating candidate passwords and comparing them with targeted sets. Therefore, these data-driven models are founded on the probability distribution of passwords, in which password interpretation practices can result in different distributions. Prior works have seemingly modeled password interpretations as generally a sequence of characters [16, 21], characters with the same classes [66], or external natural-language-based words [59], which are all password-irrelevant knowledge. Users could have their own tricks when creating passwords, e.g., leet patterns like 4ever (forever), giving rise to memorable (and predictable) units [15, 31, 69] and enabling users to create an easy way to remember passwords while still meet the requirement of "*strong*" passwords. Typical password compositions have such peculiarities that need to be taken into account in developing data-driven models.

When passwords are modeled as generic text sequences by data-driven models, it becomes hard to find the appropriate composition units of passwords. The original whole-string Markov model [45] and neural-network-based FLA (Fast, Lean and Accurate) model [43] basically adopt a *character-level* model, which regards a single character as a basic unit to compose a password. However, organizing passwords by characters might be too fine-grained. In contrast, the template-based Probabilistic Context-free Grammars (PCFG) model [66] divides a password into several units based on character classes, which sequentially used when composing a template. For instance, the template for "lastforever" is L_{11} [1], which is only related to character classes. Modeling passwords by such templates might be too coarse-grained. In a word, these models capture either *character-level* or *template-level* knowledge, leading to the granularity of password composition representations either too fine-grained or coarse-grained.

To mitigate the problem of password composition granularity, extensive prior works [35, 42, 59] are always based on hand-craft rules or password-irrelevant knowledge (i.e., words) to optimize password representations. Ma et al. [42] proposed a variable order Markov model (termed Backoff), which regards the longest character sequences whose frequency is above a threshold as a

[1] L refers to letters. In addition, D and S refer to digits and symbols, respectively

unit and gradually falls back. Veras et al. [59] employed natural-language-based techniques to segment passwords into small and meaningful units and proposed a fine-grained PCFG model (termed Semantic_PCFG). Despite the fact that these aforementioned studies mitigate the granularity problem to some extent, they suffer from the automatic extraction of refined composition representations, for that passwords exhibit their internal and refined patterns that are different from natural languages [56].

To investigate refined password composition representations, we introduce the concept of *chunk*, which is a sequence of related characters that appear together frequently to compose a password. The concept here is password-relevant obtained from the internal statistics of password datasets and intuitively similar to the *chunking* process [44] in psychology [2]. For example, the password "p@ssw0rd4ever" could be divided into two chunks (p@ssw0rd, 4ever), which strike a balance between the granularity of characters and templates.

In this work, we take a step to model passwords via a refined composition of chunks that are extracted from the internal statistics of datasets and apply chunks to *chunk-level* guessing models to understand their security impacts. We propose a password-specific segmentation method (termed *PwdSegment*) that can learn chunks insides passwords and build three *chunk-level* guessing models adopted from the Markov, PCFG and FLA models. The *PwdSegment* method extends the Byte-Pair-Encoding (BPE) algorithm [25, 48, 52] and allows passwords to be automatically interpreted into several chunks. Based on the resulting chunks, we build a *chunk-level* Markov model upon Backoff, which we call CKL_Backoff, a *chunk-level* PCFG model (termed CKL_PCFG) and a *chunk-level* neural-network-based model (termed CKL_FLA). We apply the various average length of a resulting chunk vocabulary to these models. Particularly, we design and build CKL_Backoff and CKL_FLA by replacing characters with shorter chunks (e.g. average length of 1.8), CKL_PCFG is built with the novel template consisting of longer chunks (e.g. average length of 4.5). The empirical evaluation shows that all three *chunk-level* models have substantially higher guessing performance than state-of-the-art models.

Based on these efficient attacks, we quantitatively investigate the impacts between several security-related factors and password strength. Previous works [33, 55] generally identified the length and the number of character classes as important factors affecting security. In our study, we find that the frequency of chunks in a password is a stronger indicator for password vulnerability than the lack of character class diversity (i.e., passwords are weaker when composed of high-frequency chunks), despite the fact that the length is still the most influential. Given the risks posed by *chunk-level* guessing attacks, we design and implement a client-side password strength meter by compressing CKL_PCFG to 5.0 MB for a real-time feedback, which can remind common chunks in a generated password and identify more unsafe passwords than other widely-used meters (e.g., the FLA meter [43]). Besides, our meter provides the interface for developers to conveniently integrate into password managers [1–3].

We summarize our main contributions as follows.

- We investigate and evaluate a refined password composition unit (i.e., chunk) by an automatic password-specific parser *PwdSegment*, demonstrating that previously identified *characters, templates or words* are less effective in modeling password guessing than chunks.
- We propose a *chunk-level* Backoff, PCFG and FLA guessing model. Through an extensive evaluation, we find that these models outperform the state-of-the-art models by an average of 5.7%, 51.2% and 41.9%, respectively, demonstrating significant vulnerability with passwords using common chunks.
- We design and implement a client-side password strength meter based on the compressed CKL_PCFG model to mitigate the risks from *chunk-level* attacks. Our meter can remind users that commonly-used chunks are vulnerable, which is an underestimated threat in security.

Roadmap. In Section 2, we present the threat model and password *chunks*. We elaborate on the design of CKL_Backoff, CKL_PCFG and CKL_FLA in Section 3.1, 3.2 and 3.3, respectively. In Section 4, we evaluate these *chunk-level* models' guessing performance. Afterwards, we present security recommendations and applications in Section 5. In Section 6, we discuss practical values. In Section 7, we introduce related works. Section 8 summarizes this paper.

2 PRELIMINARIES AND PASSWORD CHUNKS

We explicate the threat model and datasets used in this paper. Then we elaborate on the extraction of chunks along with their properties.

2.1 Threat Model

The objective of guessing models is to quantify how many guesses it would take for successfully cracking a password, including mostly but not limited to online targeted attacks [64] or offline trawling attacks [20, 42, 43, 47, 58, 62, 66]). In this paper, we refer most of the related works and mainly focus on offline trawling attacks as the threat model, while online targeted attacks are outside the scope of this work since the offline password guessing is common in digital forensics [12, 54].

In offline trawling guessing attacks, an attacker attempts to recover passwords under a larger number of guesses, e.g., 10^{14} [24, 55] as the maximum guesses, because it will ensure a practical attacker scenario with available computing power. A far large number of guesses, e.g., 10^{20}, usually only has its theoretical significance but is limited with practical values. Particularly, offline attacks generate these candidate passwords in descending probability order based on models. These candidate passwords are then used to be hashed by using the hashing algorithm (e.g., MD5, Argon2) to compare with the cipher-text password, in which the matching probability is the cracking rate.

2.2 Datasets

This section introduces six leaked datasets used in our paper with over 250 million plain-text passwords, including three from English sites and three from Chinese sites (shown in Table 1). They were disclosed publicly on the Internet and have been extensively used in trawling offline password models [26, 38, 61, 68]. **Rockyou** [5] is

[2] *Chunking* refers to the process of taking individual pieces of information and grouping them into a meaningful whole for a sound memory [44].

Table 1: Summary of password corpus used in this paper; The C_v describes the degree of dataset dispersion.

Region	Datasets (C_v)	Year	Account Valid (Removed%)	Account Unique	Account Length ≥ 16
English	Rockyou (0.27)	2009	32,584,165 (0.06%)	14,270,373	250,242
	Neopets (0.29)	2016	67,672,205 (1.58%)	27,474,769	1,081,602
	Cit0day (0.33)	2020	86,835,796 (0.00%)	40,589,949	1,869,877
*English			187,092,166 (0.01%)	72,095,944	3,201,721
Chinese	CSDN (0.20)	2011	6,425,243 (0.01%)	4,031,226	68,817
	Youku (0.21)	2016	47,607,615 (0.08%)	47,462,025	395,701
	178 (0.22)	2011	9,071,979 (0.09%)	3,461,974	7,431
*Chinese			63,104,837 (0.07%)	54,955,225	471,949
Total			250,197,003 (0.45%)	127,051,169	3,673,670

a popular game information website; **Neopets** [8] is a famous online pet information website. **Cit0day** is a multi-language dataset, in which most of them are from English, offering access to usernames and email services for thousands of websites. The Cit0day data breach claimed that up to 226 million usernames and passwords were leaked, yet we collected around 80 million plain-text passwords in total. **CSDN** [7] is a Chinese programming forum. **Youku** [9] provides video services. **178** [6] is a website providing game information. Most users of CSDN, Youku and 178 are Chinese speaking users, while users of Rockyou, Neopets and Cit0day are mostly English speaking. To obtain valid passwords, we remove non-ASCII passwords and too long passwords with more than 32 characters [43, 63]. Such unusually long strings are unlikely generated by users or password managers due to their default settings of 12, 16 or 20 characters in most of the managers (e.g., LastPass [3], Dashlane [2], or 1Password [1]). Moreover, they are often beyond the scope of attackers who care about cracking efficiency [12] (for completeness, we evaluate these longer passwords in Appendix A.1). We count the complexity statistics of each dataset by the coefficient of variation (C_v) [23], whose value is proportional to the degree of dataset dispersion. We observe from Table 1 that newer datasets become more and more discrete, evidenced in that Cit0day is the most discrete. We choose these diverse sets (three recent leaked datasets and the other three datasets are a bit old) of different eras to observe the temporal change in datasets. Our research in this paper only presents the aggregated statistical information rather than individual features for the requirement of ethical practice.

2.3 PwdSegment: A Password-Specific Segmentation Method

We present a password-specific segmentation method (termed *PwdSegment*) to interpret a password into several chunks. Conceptually, *PwdSegment* trains a Byte-Pair-Encoding (BPE) algorithm based on training data of plain-text passwords for obtaining chunk vocabularies. This principle is referred to the parser tool *WordSegment* [27] of training an n-gram model based on a trillion of the natural-language corpus. The BPE algorithm, originally proposed in 1994 as a data compression technique [25], is widely used in machine translation for the task of subword segmentation (e.g., the RoBERTa model [39] proposed by Facebook and GPT-2 model [50] proposed by Open AI.), which keeps the frequent words intact while splitting the rare ones into multiple units. The raw BPE algorithm first trains on the plain-text corpus. Then, it iteratively merges the most frequent pair of tokens with a single, new (i.e., unseen) token, which constitutes the subword (i.e., chunk) vocabulary. Each *merge* operation produces a new chunk by replacing the most frequent pair of characters or character sequences (e.g., "w", "0") with a new, unused subword (e.g., "w0"). The *merge* operation is repeated for a specified number of times as configured before (i.e., a hyper-parameter), resulting in a chunk vocabulary of proportional size.

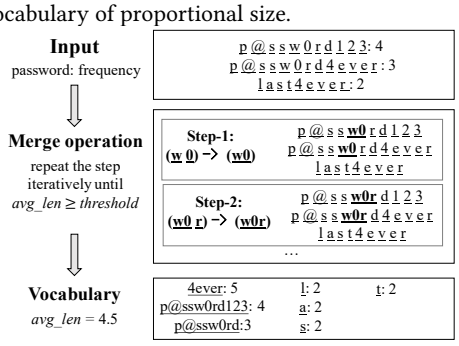


Figure 1: An example of *PwdSegment*

PwdSegment extends the BPE algorithm by using the configurable parameter of average length (for short, *avg_len*) of chunk vocabularies to replace the number of the *merge* operations. *PwdSegment* counts all pairs of characters and stops the *merge* process early when the *avg_len* of the resulting chunk vocabulary equals or exceeds the threshold length we set. Compared with the number of *merge* operations, *PwdSegment* could be parameterized with a threshold *avg_len* to tune the segmentation result with different granularity more conveniently, where a longer *avg_len* leads to a more coarse-grained result. Besides, the *avg_len* can describe the length nature of a chunk vocabulary, not as confusing as the number of *merge* operation. We use an example (shown in Figure 1) to illustrate the workflow of *PwdSegment* as follows.

(1) **Setup:** Prepare a training set of plain-text passwords and configure the expected *avg_len* of the resulting chunk vocabulary.
(2) **Input:** Count the number of passwords occurred in the training set and split passwords into character sequences. For instance, the password "p@ssw0rd123" appears 4 times, we label it with "p @ s s w 0 r d 1 2 3 : 4"
(3) **Merge operation:** Merge character pairs iteratively in descending order of frequencies. In step-1, the character pair "w 0", whose frequency is the highest (appears 7 times), is merged into w0; Note that, "p @" also appears 7 times, *PwdSegment* chooses "w 0" based on dictionary sequences, then the input is updated accordingly (shown in Merge operation of Figure 1); In the step-2, the top character pair is "w0 r", which also appears 7 times, is merged into w0r; Then the next step-3 follows the same practices and merges "w0r d" into w0rd. We detail the specific changes of each merge operation in Appendix A.2.
(4) **Generate a chunk vocabulary:** Repeat the procedure above until the *ave_len* of resulting chunks is equal to or greater than the threshold, or all character pairs have the same frequency. The resulting chunk vocabulary consists of either characters or chunks. Finally, *PwdSegment* segments passwords based on the chunk vocabulary, e.g., the "p@ssw0rd4ever" could be interpreted as "p@ssw0rd, 4ever".

Figure 2: Top-150 password chunks and natural-language words. Passwords exhibit patterns that are irrelevant to words. We colour these patterns with different colours, i.e., leet (4ever), syllables (ing), keyboard (123456789), dates (1995) and words (the) are coloured with red, orange, purple, deep-blue and green, respectively. The chunks that does not belong to patterns above are coloured with grey.

2.4 Property Analysis of Resulting Chunks

2.4.1 Difference between password chunks and natural-language-based words. To further compare their differences, we list the top-150 *chunks* and words. For chunks, we select the coarse-grained resulting chunk vocabulary with the longer *avg_len* threshold of 4.5, since most of chunks (around 50%) in fine-grained resulting chunks (i.e., *avg_len of 1.8*) are with only one or two characters. For natural-language-based words, we extract words from Corpus of Contemporary American English (COCA) [18], which is a large and up-to-date English corpus. Figure 2 shows results after removing single or two characters for cleaner presentation, from which we have the following observations:

- The password chunks exhibit a clear difference from those natural-language-based words. Digits contribute a lot to password chunks, while letters are the only character class for composing a word. Except for words, we find that chunks show other memory patterns. For example, **leet** [3] (4ever, iloveu), **syllable** [4] (ing, wang,), **keyboard** (123456789, qwerty), and **date** (2015).
- The number of pattern chunks that exactly matches (a full matching) these leet, syllable, keyboard and date patterns take up to an average of 0.10% (31,899) , 2.27% (546,284), 2.03% (439,230) and 1.21% (195,382) across four datasets respectively, suggesting that it is not enough only to consider natural-language-based words [36, 71] for splitting passwords accurately. Further, The top-N **leet** chunks (shown in Table 2) show that users prefer using "4" to replace "for", evidence in 4ever or just4you, which deserves attention for this replacing rules produce passwords much weaker than a typical letter-digit combination. Besides, for top-10 **syllable** chunks (shown in Table 3), the widely-used syllable chunks are mostly prepositions of an or in, or word suffixes for `English` chunks, and pinyins of Chinese surnames (li or wang) for `Chinese` chunks. Notably, the Chinese family names are often adopted (e.g., the top-1 li is the most commonly used Chinese family name). For other popular chunks that do not exhibit patterns above (e.g., 001 or 777), we argue that they could behave as the stopwords among passwords due to their high frequency. Besides, the newer dataset `Cit0day` includes more date chunks.

[3]Leet refers to chunks with commonly-used transformation rules, including shape similarity, e,g,, p@ssw0rd (password), and sound pronunciations, e.g., 4ever (forever), making chunks easier to remember.

[4]Syllables refer to the combinations of English affixes and Chinese pinyins, where the English affix is a morpheme attached to a word stem to form a new word or word form. (e.g., ing) and the Chinese pinyins are the pronunciation of Chinese characters (e.g., wang, ao)

Table 2: Top-N leet chunks. Note that the leet chunks from `CSDN` only contains 181 unique ones. We use "–" when the rank is larger than 181.

Rank	Rockyou		Cit0day		CSDN		Youku	
1	**4ever**	16,787	**4ever**	1,023	p@ssw0rd	356	**4ever**	205
2	love4ever	1,486	4me2	71	P@ssw0rd	353	l0ve	69
3	2cute4u	1,145	s4me	67	**4ever**	289	x1ao	31
4	4EVER	1,105	l0ve	61	l0ve	71	l0ng	20
5	2hot4u	949	w00d	54	w0rd	30	a1a2a3	17
6	sk8er	811	l0v3	54	just4you	26	5ai1	12
7	l0ve	764	w0rd	44	il0ve	19	il0ve	10
8	il0ve	687	4Ever	42	p@ss	18	s0ng	10
9	l0v3	534	P@ssw0rd	40	pa$$w0rd	16	P@ssw0rd	10
10	love4u	528	L0ve	39	P@ss	16	y@ng	9
134	L0ve	80	f4be	15	**P@ssw0**	4	9d6f	3
181	**passw0r**	67	p@ss	14	l4s9	1	1e35	3
212	**love4e**	51	7a1b	13	–	–	ce1c	3
461	laur3n	35	**passw0r**	10	–	–	sa1s	2
Total (%)	0.1603		0.1921		0.0156		0.0202	

- For the semantics in chunks, we argue that chunks could behave as the basic composition units in passwords, which should be a novel language with its own unique grammars. Still, we notice something interesting that around dozens of chunks are incomplete (e.g., love4e, passw0r) in recognized leet chunks (shown in Table 2). Actually, the next part of "love4e" and "passw0r" may not be "ver" or "d" explicitly. For example, "love4eva", "love4emily" or "love4every1", which could be split into "love4e" and "(e)va, (e)mily, (e)very1", also appear frequently. We also observe that there are a dozen of chunks like passw0rt or passw0rld. This case is similar and possibly acceptable with the word "password", which consists of "pass" and "word" in English grammar. As a result, these incomplete chunks should not be simply regarded as an error, but actually also behave as a composition unit that reflects the peculiarities of the password language, which is still unfamiliar to us. We argue that *PwdSegment* can automatically capture and generalize chunk vocabularies with unfamiliar semantics with a parameterized *avg_len*.

We detail the specific technique of identifying these memory patterns in Appendix A.3. We also show the results of a fine-grained resulting chunk vocabulary (i.e., avg_len of 1.8) in Appendix A.4, whose findings is almost the same as concluded above. Notably, a fine-grained chunk vocabulary includes more **syllables** (e.g., ing).

2.4.2 Chunks' Part-of-Speech distribution. To further investigate the difference between chunks and words, we compare the distribution of Part-of-Speech (POS) between password letters (i.e.,

Table 3: Top-10 syllable chunks.

Rank	Rockyou		Cit0day		CSDN		Youku	
1	**an**	10,614	**in**	2,940	xiazhili	3,649	**li**	2,657
2	**in**	10,341	er	2,602	**li**	3,555	yu	2,211
3	er	9,421	**an**	2,391	yu	2,952	wei	1,823
4	ka	8,520	ka	2,220	wang	2,875	lin	1,744
5	**ing**	7,781	ma	1,855	wei	2,737	yang	1,721
6	al	7,167	en	1,796	lin	2,680	wang	1,718
7	cute	6,947	ed	1,699	yang	2,538	jie	1,624
8	za	6,831	**ing**	1,671	feng	2,520	ai	1,602
9	ma	6,786	za	1,652	chen	2,498	chen	1,589
10	you	6,335	ca	1,579	hua	2,347	jun	1,532
Total (%)	2.2401		1.1456		3.4833		2.2138	

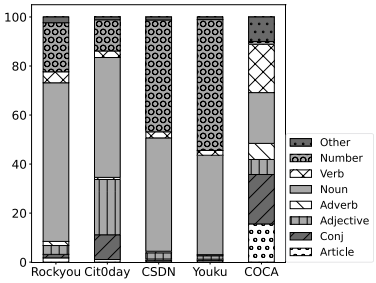


Figure 3: Distribution of Part-of-Speech between passwords and words in natural languages. The *other* includes all other types that do not belong to any of given categories above.

four typical password datasets) and natural-language words (i.e., the COCA [18]) using the toolkit of NLTK [10] with multiple PoS libraries. We conclude the result shown in Figure 3 as follows: contrary to the natural-language text, passwords are more likely to contain nouns but with much fewer verbs and adverbs, reflecting that users rarely use verbs to create passwords. Besides, we note that `Cit0day` includes more adjectives than other datasets, indicating that users are prone to use adjectives to create their passwords recently, possibly because adjectives are suitable to express their emotions. We further investigate that the top three adjectives are "new", "big", and "happy", which reside largely in "newyork", "bigdog", and "happy123".

2.4.3 Chunks follow Zipf's law. Wang et al. [60] proposed that the distribution of passwords follows a natural law of Zipf's law, which states that the frequency of a password is inversely proportional to its rank. There are two variants of Zipf's law for passwords: PDF-Zipf and CDF-Zipf models. Password frequencies follow both the PDF-Zipf model if we drop the tail passwords (e.g., passwords that appeared less than five times) and the CDF-Zipf model with the entire passwords. The *Kolmogorov-Smirnov Test* [46, 60] (for short, KS test "D_{KS}") measures how close the theoretical Zipf's distribution is with the real distribution. The smaller the "D_{KS}" is, the better the fit with the theoretic distribution. We conclude that the chunks' frequencies are also well modeled by PDF and CDF-Zipfs' law according to the sufficiently low values of KS tests (shown in Table 4).

3 CHUNK-LEVEL GUESSING MODELS

To investigate the password security under chunk representations, we propose three *chunk-level* guessing models adopted from the *whole-string* Markov model, *template-based* PCFG model, and *neural-network-based* FLA model. In this section, we introduce the design and implementation of these three *chunk-level* models.

Table 4: Fitting results of PDF and CDF-Zipfs' law.

avg_len	model	Statistics	Rockyou	Cit0day	CSDN	Youku
1.8	PDF	Coverage	99.9%	100.0%	100.0%	100.0%
		D_{KS}	0.1059	**0.0462**	0.2875	0.2005
	CDF	Coverage′	100%	100%	100%	100%
		D_{KS}′	**0.0663**	0.0879	0.0734	0.0828
4.5	PDF	Coverage	99.5%	29.8%	73.8%	30.4%
		D_{KS}	**0.0597**	0.0930	0.1000	0.0896
	CDF	Coverage′	100%	100%	100%	100%
		D_{KS}′	0.0848	0.1174	**0.0781**	0.1353

3.1 Whole-String Markov Models

Originally, Narayanan and Shmatikov [45] applied the context-dependent whole-string Markov model (also call *n-gram*) to guess passwords. An n-gram model, which is also known as a Markov model of order $n-1$, assumes that the current character is only dependent on the prior $n-1$ characters. For example, a 3-gram model (order-2 Markov) requires two prior characters to determine the next character. Formally, we denote the probability of the character c_l following the string $c_1c_2...c_{l-1}$ as follows:

$$P(c_l|c_1c_2...c_{l-1}) = \frac{N(c_1c_2...c_l)}{N(c_1c_2...c_{l-1})}$$

where $N(c_1c_2...c_l)$ refers to the number of a continuous character string $c_1c_2...c_l$ in a training dataset. Then, the probability of a password $c_1c_2...c_l$ of an n-gram model (Markov model of order $n-1$) is denoted by the product of transition probabilities as follows:

$$P(c_1c_2...c_l) = \prod_{i=1}^{l+1} P(c_i|c_{i-n+1}...c_{i-1})$$

where the c_i refers to the ith character in a password. When $i \leq 0$ or $i > l$, then the c_i (i.e., c_0, c_{l+1}) refers to a symbol that does not exist in the dataset, and is used to denote the start or the end of the passwords.

3.1.1 Traditional character-level backoff models. An *n−gram* model, especially when n is large, faces the data sparsity problem that many transition probabilities for given characters are 0. To address the issues of sparsity and order selection, Ma et al. [42] proposed a variable order Markov model for password cracking based on Katz's Backoff model [32]. The Backoff model adds the smoothing technique and introduces a fallback mechanism to estimate the next character accurately. For example, if the prefix is "p@ssw0r", using the whole prefix to determine the next character could be more accurate than using only "w0r". If a prefix rarely appears (e.g., "a#!Dce"), Backoff triggers the fallback mechanism and uses only a shorter prefix (e.g.,"ce") to estimate the next character. The Backoff model chooses the longest character prefix whose count is above the threshold ϕ to determine the next character. There are two cases to compute the transition probability from $c_1c_2...c_{l-1}$ to $c_1c_2...c_l$: In case one, $c_1c_2...c_{l-1}$'s count is above the threshold ϕ, then the probability is simply $\frac{N(c_1c_2...c_l)}{N(c_1c_2...c_{l-1})}$; In case two, when the $c_1c_2...c_{l-1}$'s count is below the threshold, then we look for a shorter prefix to determine the next character by subtracting one character

iteratively until a shorter prefix's count is above the threshold. Finally, the transition probability of using a shorter prefix should be normalized since the go back mechanism would increase the probability.

3.1.2 Chunk-level whole-string Backoff models. We design a *chunk-level* guessing model, CKL_Backoff, based on the Backoff model since it adds the smoothing technique to reduce the sparsity problems and order selection issues. One main difference between CKL_Backoff and Backoff is that CKL_Backoff uses a chunk to replace a character as the basic unit to calculate transition probabilities. To capture chunks, we build the CKL_Backoff model with the resulting chunk vocabulary of *PwdSegment* with a short *avg_len* trained on training sets. Another difference is that our CKL_Backoff model limits the number of chunks considered during the fallback mechanism to a predefined threshold to determine the next chunk; in contrast, the *character-level* Backoff chooses the longest character prefix in a password. This is for that a prefix of the max number of chunk combinations is sufficiently enough compared with that of the longest characters to achieve a successful guessing model, based on the fact that the length of chunks is longer than that of characters. Besides, the closer prefix could have a greater effect for accurately estimating the next chunks. Moreover, the longest prefix of chunks would induce exponential storage space. The *chunk-level* model has much more combinations of chunks (i.e., more transition probabilities) than a *character-level* one due to the exponential number of chunks (e.g., the vocabulary with the *avg_len* 1.8 in training sets of Rockyou and CSDN is 466 and 935, respectively) compared with that of characters (e.g., 94).

Given a threshold ϕ and the max prefix size of χ chunks, let ck_i denote the i-th chunk that composes a password, and let the $\lambda_{i,j}$ denote the substring from ck_i to ck_j (including ck_i and ck_j) in a password $ck_1ck_2ck_3...ck_l$. Then we define the probability of a password composed by more than one chunk in CKL_Backoff as

$$P(ck_1...ck_l) = \prod_{i=1}^{l+1} P(ck_i|\lambda_{max(0,i-\chi),i-1})$$

where

$$P(ck_{j+1}|\lambda_{i,j}) = \begin{cases} \frac{N(\lambda_{i,j+1})}{N(\lambda_{i,j})}, & if\ N(\lambda_{i,j+1}) \geq \phi, \\ P(ck_{j+1}|\lambda_{i+1,j})\omega(\lambda_{i,j}), & otherwise \end{cases}$$

and

$$\omega(\lambda_{i,j}) = \sum_{ck:N(\lambda_{i,j},ck)\geq\phi} \frac{N(\lambda_{i,j},ck)}{N(\lambda_{i,j})}, \quad 0 \leq i < j <= l$$

Same as the *character-level* model, each password is considered to start with a start symbol ck_0 and an end symbol ck_{l+1}. Below, we give a concrete example based on the result of *PwdSegment* with a shorter *avg_len* (i.e., 1.8) to illustrate the process of CKL_Backoff.

Example-1: *"p@ssw0rd4ever"* → p @ s sw 0 r d 4 ever
Let the threshold value ϕ be 10 and max prefix size be as large as 7 for an example. Then CKL_Backoff chooses the prefix of 7-chunks to estimate the next chunk, once the 7-chunks' count is less than 10, CKL_Backoff would go back to a shorter prefix. Suppose that the count of max prefix is all larger than the threshold 10 except the $\lambda_{2,8}$, which is the max prefix to estimate the 9-th chunk of ever. CKL_Backoff adopts the direct prefix of 7-chunks to estimate all other chunks except the 9-th chunk ever, whose transition probability's calculation should trigger a go-back mechanism by decreasing 1 iteratively from the current max size 7-chunks. Suppose that the count of a shorter prefix 6−*chunks* ($\lambda_{3,8}$) for 9-th ever is larger than 10, then the probability of "p@ssw0rd4ever" is deduced as follows:

$$\begin{aligned} P(p@ssw0rd4ever) = {} & P(p|ck_0) \times P(@|ck_0p)... \times P(4|\lambda_{1,7}) \\ & \times P(ever|\lambda_{3,8}) \times \omega(\lambda_{2,8}) \times P(ck_{10}|\lambda_{6,9}) \end{aligned}$$

where

$$\omega(\lambda_{2,8}) = \sum_{ck:N(\lambda_{2,8},ck)\geq 10} \frac{N(\lambda_{2,8},ck)}{N(\lambda_{2,8})}$$

Parameter selection. For the threshold ϕ, we follow the setting of 10 as recommended by prior work [42]. As for the max n-grams, we use 5, namely, the longest prefix is 5-chunks to estimate the next chunk due to the trade-off between a runnable storage and efficiency. For the *avg_len* of *PwdSegment*, we assign a short *avg_len* strategy, i.e., the *avg_len* shorter than 1.8, for the following reason. CKL_Backoff is a context-relevant model with a strong ability of capturing the context relevance (i.e., chunk combinations), we therefore choose the parameter within a shorter *avg_len* range for more chunk combinations. By contrast, the number of chunk combinations between longer chunks will be greatly reduced, limiting the model's ability to capture context. We discuss this trade-off using more experimental results in Section 4.1.3.

3.2 Template-Based PCFG Models

Weir et al. [66] originally extended the context-independent PCFG model for guessing passwords effectively. In the original template-based PCFG model, one converts a password into a template based on character classes including letters (L), digits (D) and symbols (S). Those consecutive characters that belong to the same category are grouped into a unit (i.e., a tag), which sequentially constitutes the template. We abstract the password probability formula as follows:

$$P(c_1c_2...c_l) = P(tag_1tag_2...tag_l) * \prod_{i=1}^{l} P(pwd_i|tag_i)$$

The the password "p@ssw0rd4ever" as an example, the probability is modeled as:

$$\begin{aligned} P(p@ssw0rd4ever) = {} & P(L_1S_1L_3D_1L_2D_1L_4) \times P(p|L_1) \\ & \times P(@|S_1) \times P(ssw|L_3) \times P(0|D_1) \\ & \times P(rd|L_2) \times P(4|D_1) \times P(ever|L_4) \end{aligned}$$

where $P(L_1S_1L_3D_1L_8S_1D_1)$ denotes the probability of a template, $P(p|L_1)$ describes the probability that "p" appears in letter instantiations (i.e., the actual letter combinations) with length 1, similarly, $P(@|S_1)$ refers to the probability that "@" appears in symbol instantiations of length 1, and applies for the rest case. In [66], these instantiations' probabilities are obtained from training sets except that the letter instantiations' probabilities are directly from external dictionaries. Later, Ma et al. [42] proposed to assign all probabilities

totally based on the counted occurrences in training sets. After training, the model contains these grammars and their probabilities, which is used to generate candidate passwords for comparing with the target dataset. The model substitutes the template tags (e.g., D_1) using their actual character combinations and outputs the candidate passwords in descending order of probabilities.

3.2.1 Chunk-level template-based PCFG models. Our *chunk-level* PCFG model constructs the templates based on the resulting chunks of *PwdSegment*. The major differences between CKL_PCFG and original PCFG reside in the tag of a template, which are explained as follows:

- **L, U, D, S**: These denotes those tags in the template which are composed of merely single type of characters including the *lowercase letters, uppercase letters, digits and symbols* respectively.
- **DM**: This denotes those tags which are composed of exactly two types of characters, which are referred to as **D**ouble **M**ixed type (e.g., 4ever).
- **TM**: This denotes those tags which are composed of **T**hree **M**ixed type of characters (e.g., p@ssw0rd).
- **FM**: This denotes those tags which are composed of **F**our **M**ixed types of characters (e.g., P@$$w0RD).

Example-2: *"p@ssw0rd4ever"* → p@ssw0rd 4ever
The *"p@ssw0rd4ever"* is divided as p@ssw0rd and 4ever, of which the template is TM_8DM_5. The probability is calculated by CKL_PCFG as follows:

$$P(\text{"}p@ssw0rd4ever\text{"}) = P(TM_8DM_5) \times P(p@ssw0rd|TM_8) \times P(4ever|DM_5)$$

CKL_PCFG trains the model, i.e., the probabilities of novel templates (e.g., TM_8DM_5) and their instantiations (e.g., p@ssw0rd|TM_8) learned from training sets. Then, same as original PCFG, CKL_PCFG generates candidate passwords by substituting these tags of "L, U, D, S, DM, TM, FM" with instantiations of the same length learned from training sets. Finally, CKL_PCFG outputs the candidate passwords in descending order. Compared with the complicated template $L_1S_1L_3D_1L_2D_1L_4$ in original PCFG, the representation like TM_8DM_5 could be more suitable to describe the composition of the "p@ssw0rd4ever".

Parameter selection. There is only one hyper-parameter, i.e., the *avg_len* of *PwdSegment*, for CKL_PCFG. Given the fact that CKL_PCFG is essentially a context-independent guessing model, which means the model cannot capture the context relevance (i.e. chunk combinations) between chunks in a password, we hereby choose the parameter with longer *avg_len* ranges, i.e., longer than 3.5, to ensure a close contextual connection within chunks.

3.3 Neural-Network-Based FLA Models

Melicher et al. [43] applied the neural-network-based LSTM model (also called FLA) to the field of password guessing in 2016. Conceptually, FLA predicts the probability of the next character in a password based on previous characters, which constitute the context. The FLA model is trained to generate a password by sequentially choosing the next character with the highest transition probability given the previous character combinations. Similar to Markov models, FLA uses symbols to denote the start and ending of a password. Melicher et al. [43] focused on *character-level* models and claimed no established dictionary of words for a *word-level* FLA. Although they implement the experiments using *syllable-level* models based on 2,000 different tokens, they observed only trivial improvements due to the default dictionary.

3.3.1 Chunk-level FLA models. Similarly, the essential difference between our CKL_FLA and *character-level* FLA is that CKL_FLA replaces characters with chunks segmented by *PwdSegment* with a shorter *avg_len*. Our CKL_FLA is trained to generate the probability of the next chunk in a password given the context of previous chunks. To train a CKL_FLA model, we first encode the input of a password composed of chunks to a one-dimensional array based on the dictionary order. Afterwards, CKL_FLA converts the array to an vector by an embedding layer, in which the context is transformed into an embedding of size *embedding_dim*, with the resulting two-dimensional vector of size *context_length* × *embedding_dim*. When there are fewer chunks than the context length, we pad the input with zeros. Note, we use an embedding layer rather than a one-hot encoding layer to reduce the problem of the sparsity of the embedding vector due to the larger size of chunks. Then the embedded vector is fed into LSTM neural networks, which is exactly hidden layers. Finally, the dense layer converts the hidden layers into the output size. The output is the possible subsequent chunks with probabilities and CKL_FLA chooses the next chunk with the highest probability. To generate passwords from the model, we enumerate all passwords whose probability is above a given threshold.

Example-3: *"p@ssw0rd4ever"* → p @ s sw 0 r d 4 ever
For example (shown in Figure 4), given previous chunks of p @ s sw 0 r d 4, we first encode them to an array (e.g., [4,8,6,7,1,5,3,2]) by the dictionary order. Then we query the network for the probability of the next chunk by steps of embedding, hidden and dense layers. The network outputs that the most possible next chunk is the ever. Later, when the next chunk is the ending symbol, the network completes the password "p@ssw0rd4ever";

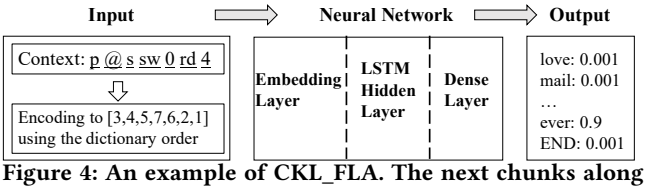


Figure 4: An example of CKL_FLA. The next chunks along with their probabilities are the output of the network.

Parameter selection. CKL_FLA is a small model with with 1,245,128 parameters. For most of hyper-parameters, we follow the settings in the *character-level* FLA model [43]. For instance, we follow the *context_length* of 10 and adopt 2 LSTM layers. We use Adam optimizer and adopt 30 training iterations for converging. Notably, we add the *embedding_layer* of 64 dimensions to embed the input context for reducing the sparsity of matrix. We choose 64 as the embedding dimension as it shows almost no performance drop compared to larger embedding sizes, which could introduce more time costs. Unlike *character-level* FLA, we use a LSTM layer with 256

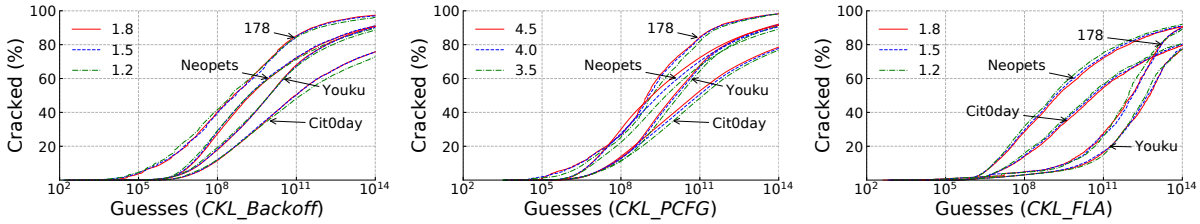


Figure 5: Impacts of the hyper-parameter *avg_len* for three *chunk-level* guessing models.

hidden-state dimensions followed by a fully connected layer with 128 dimensions, which offers the best trade-off between the time and efficiency based on preliminary experiments. For the *avg_len* of *PwdSegment*, we refer the principle in whole-string CKL_Backoff and select a shorter *avg_len* strategy (i.e., shorter than 1.8).

4 EVALUATION

4.1 Experimental Setup

4.1.1 Experimental scenarios. Our experiments adopt cross-site guessing scenarios [43, 63], i.e., we train a model using an old dataset (i.e., leaked in earlier years) and then apply the model to crack a recently leaked dataset. For `English`, we train a model using `Rockyou` to crack `Neopets` and `Cit0day`, while for `Chinese`, we train a model using `CSDN` to crack `Youku` and `178`. We refer to works [47] that remove the duplicate passwords that occur in training sets from evaluation sets to demonstrate the models' ability to generate new (or unseen), valid passwords. Besides, we randomly sample 10% of passwords in evaluation sets to ensure larger training sets and eliminate the impact of small training sets [38, 47] since that data-driven models should be theoretically guaranteed that training sets are larger than evaluation sets. We introduce the *Min_auto* strategy [58] that shows the performance of using multiple guessing methods since single models could be biased.

Kelly et al. [33] reported that the password policy of *basic16*, i.e., "Passwords must have at least 16 characters without any restrictions", is a better policy. Long passwords have been claimed as secure nowadays [30, 33, 34, 53]. Therefore, we add the long password guessing scenario to explore whether the *chunk-level* models could also improve in cracking *long* passwords. In *all* password guessing scenarios, all-length passwords are used to measure the performance, while in *long* password guessing scenarios, we extract the passwords whose length is at least 16 from datasets and then refer to these passwords as *long* passwords. We train a model using a training set of long passwords to crack an evaluation set of long passwords. For the unity of experiments, we apply the same chunk vocabularies learned from a training set of all passwords to segment both *all* and *long* passwords, in which long passwords generally imply more chunks [5]. This is also for generating high-quality chunk vocabularies due to the small number of long passwords for training a *PwdSegment*. For completeness, we also evaluate the performance of longer passwords with more than 32 characters in Appendix A.1.

4.1.2 Experimental hardware summary. We deploy CKL_Backoff and CKL_PCFG on the CPU server with Intel Xeon Silver 4210 processor and deploy CKL_FLA on the GPU server with the Nvidia GeForce GTX 2080 Ti since the neural-network models could take advantage of the parallel processing power. We adopt the Monte Carlo [20] algorithm to calculate the number of guesses for a password by enumerating all chunk combinations and selecting the combination with the highest probability. For CKL_Backoff, we refer to the previous work [20], which store no model by integrating the training process into the Monte Carlo process. When the training set is `Rockyou`, it takes around 36.0MB and 5.3MB storage space for training CKL_PCFG and CKL_FLA. For time cost, it will take around 3 days for CKL_FLA with batch size 128.

4.1.3 Experimental hyper-parameter selection. We empirically evaluate impacts of the hyper-parameter of *avg_len*, instead of hand-picking one, on *chunk-level* models by experimental scenarios of *all* password guessing described in the prior Section 4.1.1. Given the ranges analyzed in Section 3 on their principles, we experiment in CKL_Backoff and CKL_FLA with three shorter *avg_len* (i.e., 1.2, 1.5 and 1.8), CKL_PCFG with the longer *avg_len* (i.e., 3.5, 4.0 and 4.5). As shown in Figure 5, we observe that using different *avg_len* on given ranges can achieve almost the same guessing efficiency, indicating that *chunk-level* models are insensitive to hyper-parameters. We observe that CKL_Backoff achieves the best performance under *avg_len* of 1.8, CKL_FLA performs differently on four datasets, and CKL_PCFG performs the best under *avg_len* of 4.5. To balance time cost and efficiency, we settle on the manageable hyper-parameter *avg_len* of 1.8 for CKL_Backoff and CKL_FLA, the *avg_len* of 4.5 for CKL_PCFG. For example, *PwdSegment* will take 90 hours for training on `Rockyou` with the longer *avg_len* of 5.0, yet only takes 12 hours with the *avg_len* of 4.5.

To our knowledge, few scientific methods can find the best hyper-parameter. Instead, a large number of empirical experiments is a promising way. Still, rather than a hand-picking one with *avg_len*, we perform a sensitivity analysis by theoretical analysis and three groups of experiments and settle on two selected *avg_len* for these *chunk-level* models due to the trade-off between cost and efficiency.

4.2 Experimental Results

4.2.1 Results of CKL_Backoff. We report a direct comparison of the guessing performance for CKL_Backoff (the following CKL_Backoff is with the *avg_len* of 1.8) compared with three improved *character-level* Markov models as follows:

- *4-gram* [20]: The 4-gram model (i.e., the Markov model with order-3) as recommended in literature [38].
- *Backoff* [42]: The variable order Markov guessing model.
- *OMEN* [22]: The practical and fast Markov model using an Ordered Markov Enumerator (OMEN).

[5]The number of chunks in *long* passwords is averagely 7.66, compared with that of 4.49 in *all* passwords among four evaluation sets. The longer passwords with more than 32 characters contain an average of 17.23 chunks.

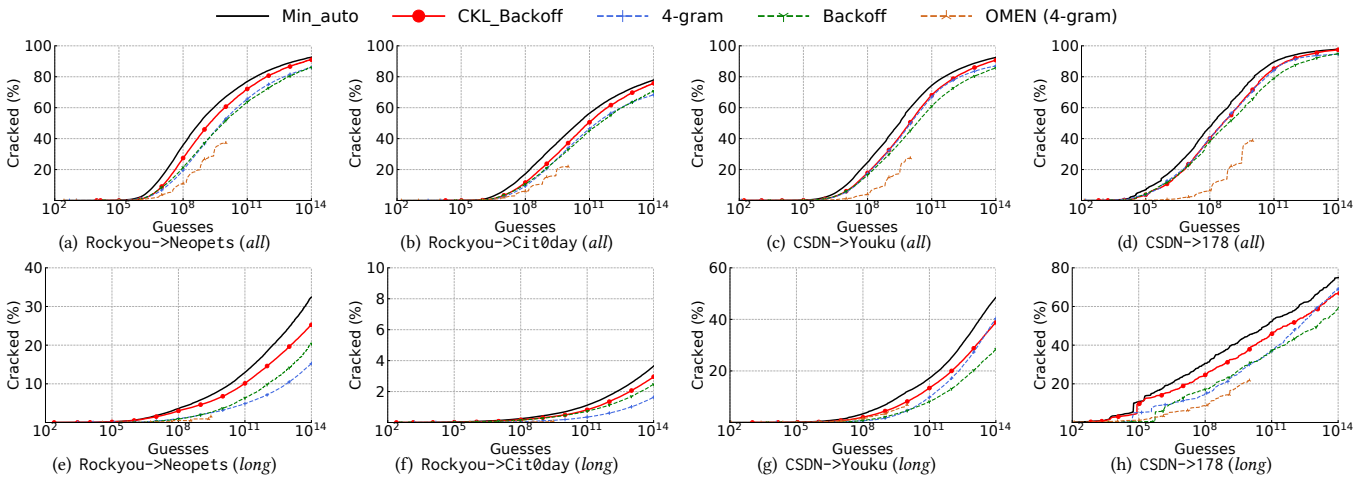


Figure 6: Guessing performance of CKL_Backoff against other state-of-the-art Markov models. The *Min_auto* represents an idealized guessing approach, in which a password is considered guessed as long as any of these guessing models cracks it.

As shown in Figure 6, CKL_Backoff outperforms the state-of-the-art Markov models in *all* password guessing scenario. CKL_Backoff manages to increase the crack rate by an average of 5.7% (from 3.0% to 11.0%) than the individual best-performing model (i.e., 4-gram) under 10^{14} guesses. When cracking *long* passwords, we find that CKL_Backoff achieves an improvement in `English` long passwords, while none in `Chinese` long passwords, possibly due to the poor chunk vocabularies. In *PwdSegment*, the `CSDN` training set (6,425,243) is much smaller than `Rockyou` (32,584,165), which may result in a weaker chunk vocabulary.

These *character-level* Markov models are too fine-grained (i.e., passwords are created characters by characters), which could lead to inefficiency. CKL_Backoff mitigates this problem by leveraging the chunk knowledge, which makes it easier to guess passwords with high-frequency chunks. Still, *Min_auto* outperforms individual models by a notable margin, suggesting that using multiple guessing methods should still be better than using any single method for accurate strength estimation, even though that CKL_Backoff outperforms other approaches individually.

4.2.2 Results of CKL_PCFG. Here, we evaluate CKL_PCFG (the following CKL_PCFG is with the *avg_len* of 4.5) compared with three improved PCFG models as follows:

- *Semantic_PCFG* [59]: The optimized PCFG model using the natural-language-based word knowledge.
- *V4.1_PCFG* [65]: The latest PCFG model using hand-craft rules.
- *Hybrid_PCFG* [35]: An improved PCFG model which adds the hybrid templates to model the whole password.

As shown in Figure 7, CKL_PCFG significantly outperforms all state-of-the-art models in *all* and *long* password guessing scenarios. Particularly, the CKL_PCFG model outperforms the best-performing competitor (i.e., Semantic_PCFG) by an average of 51.2% (from 11.5% to 131.9%) and 139.9% (from 85.7% to 214.2%) when cracking *all* and *long* passwords, respectively. In addition, we observe a remarkable improvement in `Chinese` datasets in CKL_PCFG. This could be because the improved PCFG models are oriented for `English` datasets based on English knowledge (i.e., words). CKL_PCFG bridges the gap by automatically capturing their training sets' internal and statistical knowledge, showcasing its significant generality.

4.2.3 Results of CKL_FLA. Finally, we empirically evaluate the guessing performance of CKL_FLA (the following CKL_FLA is with the *avg_len* of 1.8) compared with the *character-level* FLA model [43]. As shown in Figure 8, CKL_FLA is better at guessing *all* passwords, improving the guessing efficiency by an average of 41.9% (from 12.5% to 83.8%), and yields a similar guessing performance when cracking *long* passwords. The moderate guessing performance in cracking long passwords could be because that FLA itself can solve the problem of long-distance dependence by long short-term memory networks, while modeling chunks may over-fit. Notably, the individual CKL_FLA model performs almost the same performance as *Min_auto* when cracking *all* passwords, indicating that CKL_FLA could crack almost all passwords cracked by FLA. Still, CKL_FLA cracks other unique passwords.

4.3 Analysis of Results

4.3.1 Performance comparison across three chunk-level guessing models. Here, we make a detailed performance comparison for three *chunk-level* guessing models, i.e., CKL_Backoff, CKL_PCFG and CKL_FLA. For the absolute cracking rate, CKL_PCFG performs the best in *all* and *long* password guessing scenarios and cracks an average of 90.1% (from 78.6% to 98.2%) and 39.2% (from 5.2% to 64.9%), respectively. CKL_Backoff runs the second and cracks an average of 88.7% of *all* passwords, while CKL_FLA cracks an average of 84.5% of *all* passwords. When cracking *long* passwords, CKL_Backoff and CKL_FLA crack an average of 33.4% and 36.8% *long* passwords, respectively. For the relative improvement than best-performing competitors, CKL_PCFG also performs the best, i.e., CKL_PCFG manages to increase the cracking rate by averagely 51.2% than Semantic_PCFG, while CKL_FLA and CKL_Backoff improve their performance by 41.9% and 5.7% than FLA and 4-gram, respectively, when cracking *all* passwords. These improvements suggest that all types of data-driven password guessing models

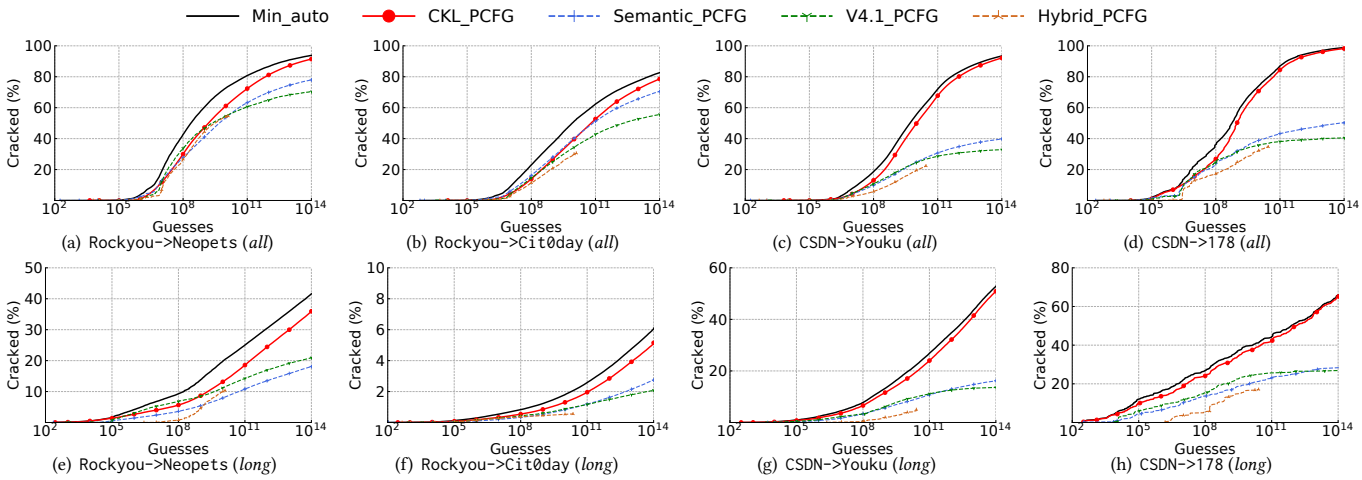


Figure 7: Guessing performance of CKL_PCFG against other state-of-the-art PCFG models.

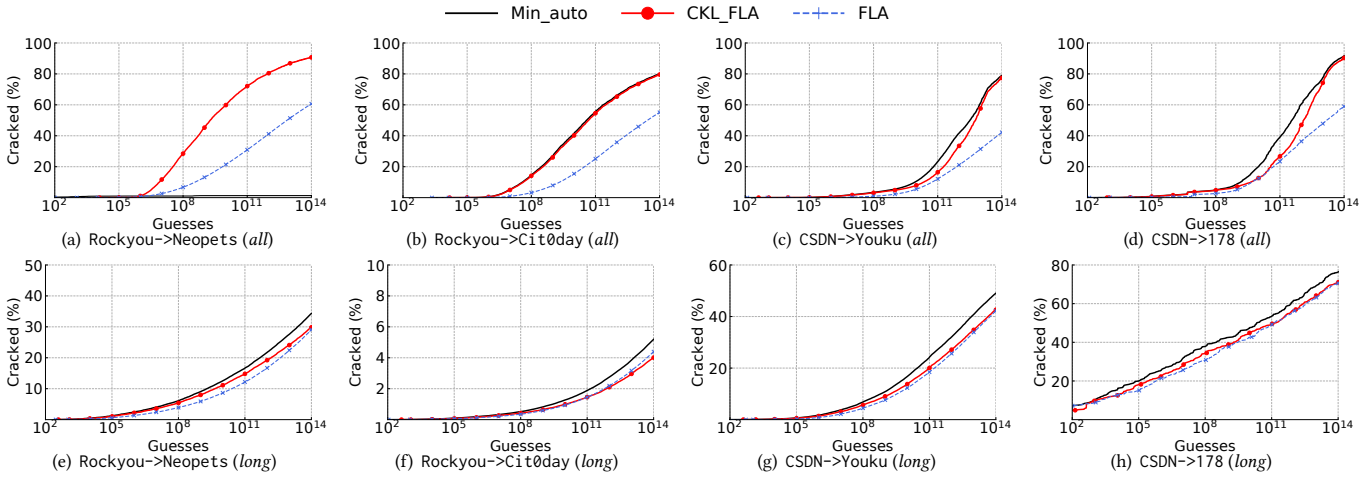


Figure 8: Guessing performance of CKL_FLA against the *character-level* FLA model.

could gain by capturing the chunk representations, which could be closer to users' creation habits.

4.3.2 Performance comparison across four evaluation sets. We note that the cracking rate in 178 is the highest (e.g., almost 100% in *all* password cracking), while that in `Cit0day` is relatively lower (e.g., less than 6% in *long* password cracking), indicating that the 178 dataset is most vulnerable and `Cit0day` is relatively secure. We speculate that this phenomenon could be caused by 178 being old and `Cit0day` being the newest. Over time, system administers design more sophisticated password creation policies to improve security. Meanwhile, users become more security-conscious to create their better passwords. Particularly, through brief experiments, we find that 178 is flooded with too many **homogeneous-passwords** (e.g., "111111111"), as counted up to 2.7% (13.1% for long passwords in 178) while these homogeneous-passwords are almost extinct in other datasets. We also find that `Cit0day` contains lots of random-seeming strings [6] compared with other `English` datasets [7]. Given the evidence that `Cit0day` is the most discrete dataset (shown in Table 1), the temporal change (i.e., the distribution of this newer dataset has changed) might explain their lower cracking rate. Still, even if the distribution of `Cit0day` has changed over time, chunks are applicable for interpreting these recent passwords. Meanwhile, *chunk-level* models achieve performance improvement. For the other two datasets whose leaked time is between 178 and `Cit0day`, we note that their cracking rates are in the middle, confirming that the leaked time of datasets has a certain impact on security.

4.3.3 Discussion of long password guessing. Although an improvement from *chunk-level* models has been shown, the improvement in long passwords is not as large as expected for that long passwords usually contain more chunks. We make a deeper discussion

[6] E.g., the top-3 in *all* and *long* passwords of `Cit0day` are "neda0409, x4ivygA51F, Groupd2013" and "h54rsjrF5J46788998, blablablablaasdasd, 4aad27137bbe266c", respectively.

[7] E.g., the top-3 in *all* and *long* passwords of `Neopets` are "ladgj, petpet12, neopets00" and "12121212121212121212, 10000000000000000000, neopets123456789", respectively.

Table 5: Cracking rates across long passwords with the different number of chunks. We combine the long passwords with less than 5 or larger than 9 chunks as a whole for their special small number. E.g., the long passwords with fewer than 5 chunks occupy around an average of 13% among four evaluation sets. The CL_B, CL_P, and CL_F represent CKL_Backoff, CKL_PCFG, and CKL_FLA, respectively.

Num (%)	Neopets (%)			Cit0day (%)			Youku (%)			178 (%)		
	CL_B	CL_P	CL_F	CL_B	CL_P	CL_F	CL_B	CL_P	CL_F	CL_B	CL_P	CL_F
≤ 5 (13)	63	77	63	46	64	50	86	93	92	70	80	66
6 (30)	28	44	30	9	16	10	41	55	43	63	73	62
7 (25)	20	30	21	8	15	9	34	50	34	68	49	69
8 (16)	13	16	13	1	2	1	13	20	12	20	36	27
≥ 9 (16)	2	2	3	0	0	0	70	10	60	8	5	4

of specific cases around long passwords with different numbers of chunks [8] and show their cracking rates under three *chunk-level* models in Table 5, from which we conclude that **CKL_PCFG achieves a higher cracking rate when cracking long passwords with fewer (i.e., fewer than 8) chunks**, while CKL_Backoff is more suitable for cracking those long passwords with more (i.e. more than 9) chunks. These findings can help us to choose an appropriate *chunk-level* model when we try to efficiently guess different types of long passwords.

4.3.4 The principle behind the improvement. The principle behind the appealing improvements could be that real-world passwords tend to cluster in a distribution. The password distribution is typically non-uniform and composed of several dense zones (i.e., some characters often co-occur) due to the heavy reuse behaviours from multiple accounts [17, 29]. The chunks, which constitute dense zones (e.g., the user habits of expression compositions #1, leet 4ever or other patterns), are still treated as independent characters in standard models. The *chunk-level* models could improve the probability of these chunks, i.e., $P(c_1c_2c_3) > P(c_1) \times P(c_2) \times P(c_3)$. The probability of chunks that constitute widespread and popular passwords is increased.

4.3.5 Limitations. First, even if chunks become the basic composition unit, it loses the property of the characters it is composed of. For example, "passw0rd" and "password" should have been regarded as similar by characters, yet they could be two chunks with no relevance, and *chunk-level* models would not be aware of their similarity. Second, Cit0day is a multi-language (most of them are English) dataset with a collection of numerous (thousands of) websites, resulting in an overall result and an ignoring result of an individual website.

5 SECURITY IMPLICATIONS AND APPLICATIONS

5.1 Security Implications

To better guide security community standards, it is critical to understand what types of passwords are hard to crack. Based on the successful attacks of *chunk-level* models, we apply them to get a quantitative understanding of how different factors influence password security. We select two widely-used factors (i.e., password length, the number of character classes) and the frequency of chunks [9] in a password and adopt the weighted Spearman's rank correlation coefficient (ρ_ω) [26] to quantitatively evaluate their correlation to password security as measured by *chunk-level* models. A higher correlation means the factor is a more reliable gauge of password strength. As shown in Table 6, we conclude that the frequency of chunks in a password is more indicative of password security than widely-recognized character class complexity (i.e., passwords are weaker when composed of commonly-used chunks), although the length is still the most influential factor [33].

Table 6: Correlations (ρ_ω) between three factors and password security by weighted Spearman correlation; The len, class and freq represent length, the number of character classes and the frequency of chunks, respectively.

Models	Neopets (ρ_ω)			Cit0day (ρ_ω)			Youku (ρ_ω)			178 (ρ_ω)		
	len	class	freq	len	class	freq	len	class	freq	len	class	freq
CKL_Backoff	0.49	0.25	0.29	0.43	0.37	0.46	0.55	0.42	0.24	0.56	0.27	0.30
CKL_PCFG	0.53	0.40	0.31	0.47	0.37	0.41	0.48	0.35	0.29	0.42	0.18	0.36
CKL_FLA	0.60	0.25	0.32	0.54	0.36	0.45	0.37	0.12	0.13	0.21	-0.14	0.15

Actionable takeaways for security recommendations. First, user-generated passwords may still be vulnerable to guessing attacks; even if they look secure or adhere to character class complexity, a proper password manager should be recommended. Second, a significant limitation for human-created password security may be due to the presence of high-frequency chunks, suggesting that in addition to just requiring more character classes or longer length, service providers must carefully consider common chunks, e.g., a human-created password like "p@ssw0rd4ever" can weaken security. Possibly, perceiving the risk element of common chunks is valuable for managers' strategy [40], which could alarm for a user-generated password containing vulnerable chunks when storing the password in the manager.

5.2 Characteristics Distribution

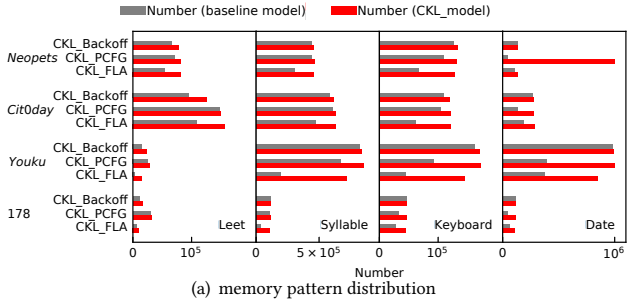
Here, we investigate some characteristics, including the memory patterns and security distribution for all cracked passwords across *chunk-level* models and their best-performing competitors. Note, the CKL_Backoff, CKL_PCFG and CKL_FLA's best-performing competitors are 4-gram, Semantic_PCFG and FLA, respectively.

5.2.1 Memory pattern distribution. To show the capability of memory pattern passwords, we compare the number of pattern passwords in all cracked passwords across different models. As detailed in Section 2.4, we mainly consider four types of patterns (i.e., leet, syllable, keyboard and date) and regard passwords that partly match the defined patterns as memory pattern passwords. We show the results in Figure 9(a), from which we conclude that without *chunk-level* models, these pattern passwords' strength can be severely overestimated.

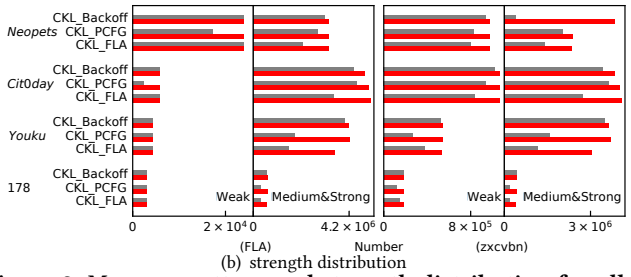
5.2.2 Security distribution. To show the security distribution for all cracked passwords across different models, we use two widely-used

[8] We adopt the longer resulting chunk vocabulary (e.g., with *avg_len* 4.5) after removing single or two characters for reducing false positive, since around 50% chunks in the shorter resulting chunk vocabulary are one or two characters.

[9] We rate the frequency of chunks as the average frequency of chunks in a password segmented by *PwdSegment* with longer results as unified. When segmenting, we adopt the maximum weight segmentation strategy.



(a) memory pattern distribution



(b) strength distribution

Figure 9: Memory pattern and strength distribution for all cracked passwords across *chunk-level* models and their best-performing competitor models. In strength distribution, the "weak", "medium", and "strong" passwords refer to those with $guesses < 10^6, 10^6 \leq guesses < 10^{14}$, and $guesses \geq 10^{14}$ evaluated by FLA and zxcvbn.

utility tools of password strength meters (FLA [43] and zxcvbn [67]) to evaluate their strength.

We classify password security strength into the "weak", "medium", and "strong" categories, using the estimated guesses for online and offline guessing [24] as boundaries. As shown in Figure 9(b), we find that the number of weak passwords cracked by these models are not much different, yet *chunk-level* model can crack more passwords with "medium" and "strong" strength, indicating our *chunk-level* models have a better advantage over their competitors with cracking strong passwords compared to weak passwords.

5.3 Practical Password Strength Meter

We enable a password strength meter based on CKL_PCFG [10], since CKL_PCFG achieves the highest cracking rate. Considering the deployment challenges in a browser, the storage of 36MB of CKL_PCFG is still too large for real-time feedback. We further compress the raw CKL_PCFG model into a manageable one with 5.0MB storage using the `g-zip` technique while the accuracy is preserved. Our CKL_PCFG meter is being hashed and deployed in the client since CKL_PCFG includes the statistical information of datasets (e.g., the template probabilities) that should be preserved from being gained by attackers. Our meter reminds users of the hazards of common chunks by employing a blocklist, which consists of the top-1000 long chunks (these top-1000 chunks already take up to 21% ~ 26% among total chunks with *avg_len* of 4.5) while reducing the weight of checking character class complexity. Figure 10 shows the interface of our meter, in which common chunks become visible (if clicked), e.g., a common chunk (4ever) is labelled as red for a warning. We refer to the literature [24] and apply three security levels, including "weak (guesses $< 10^6$), medium ($10^6 \leq$guesses$< 10^{14}$), strong (guesses$\geq 10^{14}$)" in our meter. As users who perceive vulnerability to threats tend to adopt secure practices [28, 70], we believe that reminding chunks could reduce the use of common chunks.

[10]The meter is open-sourced in https://github.com/snow0011/CKL_PSM/tree/main along with the used common chunk vocabularies.

Figure 10: Interface of our password strength meter.

Table 7: Accuracy (ρ_ω) and coverage (%) across meters; The ρ_ω is calculated by weighted Spearman correlation of our given models to evaluation sets, the Coverage (%) refers to the percentage of passwords in evaluation sets.

Meter	Neopets		Cit0day		Youku		178	
	ρ_ω	%	ρ_ω	%	ρ_ω	%	ρ_ω	%
CKL_PCFG	**0.563**	95.3	0.748	83.5	**0.310**	80.6	0.307	100.0
FLA	0.246	76.1	0.673	64.7	0.291	44.2	0.170	62.8
zxcvbn	0.496	99.4	0.756	98.0	0.306	99.1	0.405	99.3

5.3.1 Quality comparison across meters. To evaluate the quality of our meter, we follow prior two metrics of **accuracy** [26] and **unsafe errors** [43]. To obtain general results across meters in real world, we choose the experimental scenario without de-duplication between training and testing sets. For **accuracy**, we refer to practices [26] that count how meters' guesses for a given password is correlated with occurrence frequencies in an evaluation set using the weighted Spearman coefficient (ρ_ω). As shown in Table 7, our meter always achieves better accuracy than the FLA meter, and in half of the cases, it is better than zxcvbn. In `Neopets`, CKL_PCFG has a correlation of 0.563, compared with 0.246 and 0.496 in FLA and zxcvbn, respectively.

The **unsafe errors** [43] refer to rating guessable passwords as "strong" or namely, overestimation of password strength. Carnavalet et al. [19] found that many different meters give highly inconsistent feedback for the same password. Overestimation of password strength can be disastrous since the seemingly secure passwords could be actually not safe or very vulnerable. We investigate these meters' *unsafe errors* and show the average results of four evaluation sets in Table 8, from which we conclude our meter can have fewer *unsafe errors* based on the greater importance of the first column (since it refers the number of passwords whose strength is severely overestimated by other meters). For example, our meter rates 1,875 passwords as being guessable within 10^0 and 10^2 guesses, while the FLA meter rates them as larger than 10^{14} guesses. Both two metrics indicate that our meter will be better among given meters.

6 DISCUSSION

Practical values. Our study offers practical values as follows. First, by the stronger correlation between vulnerable passwords and the presence of high-frequency chunks than character class complexity, we shed light on an underestimated security factor of using

Table 8: *Unsafe errors* among meters. The first column should be more important since it refers the number of passwords whose strength is severely overestimated by other meters. For example, our CKL_PCFG meter evaluates 1,875 passwords (i.e., an average of four evaluation sets) as guessed between 10^0 and 10^2 guesses, while the FLA meter rates them as larger than 10^{14} guesses. Overestimates of strength are shown in shades of red, underestimates in purple, and accurate estimates in green. Color intensity rises with the number of passwords in a cell.

FLA	CKL_PCFG						zxcvbn	CKL_PCFG					
	$>10^0$	$>10^2$	$>10^5$	$>10^8$	$>10^{11}$	$>10^{14}$		$>10^0$	$>10^2$	$>10^5$	$>10^8$	$>10^{11}$	$>10^{14}$
$>10^0$	**0**	0	0	0	0	0	$>10^0$	**204**	36	1	0	0	0
$>10^2$	**10**	6	0	0	0	0	$>10^2$	**589**	580	636	107	20	4
$>10^5$	**82**	30	76	27	0	0	$>10^5$	**1,060**	65	1,073	1,568	457	53
$>10^8$	**324**	22	20	185	36	0	$>10^8$	**629**	3	68	629	1,165	596
$>10^{11}$	**233**	5	3	13	80	18	$>10^{11}$	**40**	0	0	4	36	277
$>10^{14}$	**1,875**	620	1,679	2,085	1,563	996	$>10^{14}$	**1**	0	0	0	0	84

common chunks. With this understanding, users would be willing to choose a password manager to create their valued passwords. Second, our password strength meter, which is open-sourced, provides a practical application with a blocklist with common chunks, and an interface for developers to conveniently integrate into password managers. Finally, *chunk-level* models provide insights into typical password compositions that can be used to more effectively guess certain complex-looking passwords (i.e., passwords meeting requirements of meters or managers like "p@ssw0rd4ever"). These insights are important in many fields (e.g., digital forensics). Considering today's challenges in digital forensics for password cracking (e.g., for law enforcement agencies to recover encrypted data of criminals), high guessing efficiency is a necessity [54], especially efficiently cracking complex-looking passwords.

Future works. In future, we would explore more *chunk-level* guessing models including the gated recurrent units (GRU) or the convolution neural networks (CNN), providing a deeper discussion of how chunks play a role in neural-network-based guessing models.

7 RELATED WORK

7.1 Password Composition Representations

The password security could be dived into a better understanding of password composition by means of breaking passwords into components and characterizing their structural patterns. Jakobsson et al. [31] firstly attempted to understand password composition focusing on the percentage of word components such as dictionary words, numbers. For long passwords, Rao et al. [51] found that users tend to create long passwords using grammatical structures and further. Booneau et al. [15] concluded that multi-word passphrases have some promise to improve security over traditional passwords. Li et al [37] constructed an empirical analysis of the Chinese user habits when creating passwords. For other composition habits, Ur et al. [57] conducted a qualitative lab study to uncover the misconceptions about weak and strong passwords, adding "!" in the end of passwords could not result in strong passwords, while users typically mistook them as strong. Recently, researchers found that centralized password reuse information [29] in password creation.

7.2 Password Guessing Models

Password cracking models are largely divided into data-driven guessing models and rule-based guessing tools (e.g., Hashcat [4]). The data-driven models are roughly divided into whole-string based and template-based models. For whole-string based models, in [45], Narayanan and Shmatikov proposed to use standard Markov modeling techniques to reduce the search size of password space. Later, Castelluccia et al [16] introduced the concept of *n-gram* to the Markov guessing models. Recently, Durmuth et al. [22] proposed an improved Markov-based password guessing model, namely *Ordered Markov Enumerator* (OMEN). Meanwhile, Melicher et al. [43] leverages the neural network to build a whole-string guessing model. From the template-based view, PCFG is a typically representative model originally proposed by Weir et al. [66]. Veras et al. [59] developed an improved PCFG with the novel templates using password-irrelevant word information. Komanduri et al. [35] considered the different classes of characters and proposed a hybrid PCFG that add the modeling of the whole password. Later, Weir et al. [65] proposed the latest version 4.1 of PCFG that adds the hand-crafted patterns (e.g., keyboard patterns). Another similar scheme is presented in [36], where Li et al. studied the security impact of personal information via PCFG models. There are also many other password cracking tools available, among which the most popular two is rule-based John the Ripper (JtR) [38], and Hashcat [4].

8 CONCLUSION

This paper takes a step towards a more refined, delicate or sophisticated way of composing passwords (i.e., chunk) to replace the conventional rough ones (i.e., character, templates or words). Particularly, we develop a password-specific segmentation method *Pwd-Segment* and propose three *chunk-level* guessing models. Through extensive evaluation, CKL_Backoff, CKL_PCFG and CKL_FLA manage to increase the guessing efficiency by an average of 5.7%, 51.2% and 41.9% than their best-performing competitor models, showing that all types of data-driven models could indeed gain by modeling the promising chunk representations. By analysing the efficient attacks, we regard the common chunks as an underestimated threat of password security, for that high-frequency chunks in a password contribute a larger vulnerability impact than the widely-used character class complexity. To mitigate these risks, we enable a client-side password strength meter based on CKL_PCFG with sub-second latency, offering warnings of common chunks for security.

ACKNOWLEDGEMENT

We thank all anonymous reviewers and Dr. Xinyi Zhang for their insightful comments. This paper is supported by NSFC (Grant NO.: U1836207) and STCSM Key Projects (Grant NO.: 21511101600). The corresponding author is Weili Han.

REFERENCES

[1] 1Password. https://1password.com/. https://1password.com/.
[2] Dashlane. https://www.dashlane.com/. https://www.dashlane.com/.
[3] LastPass. https://lastpass.com/. https://lastpass.com/.
[4] Jens, Steube Hashcat. https://hashcat.net/hashcat/,2009-., 2009-.
[5] Rockyou. https://www.rockyou.com/. https://www.rockyou.com/, 2009.
[6] 178. https://www.178.com/, 2011.
[7] CSDN. https://www.csdn.net/. http://www.csdn.net/company/about.html, 2011.
[8] Neopets. https://www.neopets.com/, 2011.
[9] Youku. https://www.youku.com/. Youku. https://www.youku.com/, 2015.
[10] NLTK, 2017. http://www.nltk.org/_modules/nltk/stem/snowball.html.
[11] English affix, 2018. https://sites.google.com/site/itskys/englsih-study/ying-yu-zhong-chang-jian-de-qian-zhui-ji-qi-shi-yi.
[12] Jeremiah Blocki, Benjamin Harsha, and Samson Zhou. On the economics of offline password cracking. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 853–871. IEEE Computer Society, 2018.
[13] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *2012 IEEE Symposium on Security and Privacy*, pages 553–567, 2012.
[14] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. Passwords and the evolution of imperfect authentication. *Commun. ACM*, 58(7):78–87, 2015.
[15] Joseph Bonneau and Ekaterina Shutova. Linguistic properties of multi-word passphrases. In Jim Blythe, Sven Dietrich, and L. Jean Camp, editors, *Financial Cryptography and Data Security - FC 2012 Workshops, USEC and WECSR 2012, Kralendijk, Bonaire, March 2, 2012, Revised Selected Papers*, volume 7398 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2012.
[16] Claude Castelluccia, Markus Dürmuth, and Daniele Perito. Adaptive password-strength meters from markov models. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012.
[17] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and Xiaofeng Wang. The tangled web of password reuse. In *Proceedings of NDSS*, 2014.
[18] M. Davies. The corpus of comtemporary american english:425 million words, 1990-persent, 2020.
[19] Xavier de Carné de Carnavalet and Mohammad Mannan. A large-scale evaluation of high-impact password strength meters. *ACM Trans. Inf. Syst. Secur.*, 18(1):1:1–1:32, 2015.
[20] Matteo Dell'Amico and Maurizio Filippone. Monte carlo strength evaluation: Fast and reliable password checking. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 158–169. ACM, 2015.
[21] Matteo Dell'Amico, Pietro Michiardi, and Yves Roudier. Password strength: An empirical analysis. In *Proceedings IEEE INFOCOM 2010*, pages 1–9. IEEE, 2010.
[22] Markus Dürmuth, Fabian Angelstorf, Claude Castelluccia, Daniele Perito, and Abdelberi Chaabane. Omen: Faster password guessing using an ordered markov enumerator. In Frank Piessens, Juan Caballero, and Nataliia Bielova, editors, *Engineering Secure Software and Systems*, pages 119–132, Cham, 2015. Springer International Publishing.
[23] Brian Everitt. *The Cambridge dictionary of statistics*. Cambridge University Press, Cambridge, UK; New York, 2002.
[24] Dinei Florêncio, Cormac Herley, and Paul C. van Oorschot. Pushing on string: the 'don't care' region of password strength. *Commun. ACM*, 59(11):66–74, 2016.
[25] Philip Gage. A new algorithm for data compression. *The C Users Journal*, 12:23–38, 02 1994.
[26] Maximilian Golla and Markus Dürmuth. On the accuracy of password strength meters. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1567–1582. ACM, 2018.
[27] Grant.Jenks. Wordsegment 1.3.1:description, 2018.
[28] Mumtaz Abdul Hameed and Nalin Asanka Gamagedara Arachchilage. On the impact of perceived vulnerability in the adoption of information systems security innovations. *CoRR*, abs/1904.08229, 2019.
[29] Weili Han, Zhigong Li, Minyue Ni, Guofei Gu, and Wenyuan Xu. Shadow attacks based on password reuses: A quantitative empirical analysis. *IEEE Trans. Dependable Sec. Comput.*, 15(2):309–320, 2018.
[30] Weili Han, Ming Xu, Junjie Zhang, Chuanwang Wang, Kai Zhang, and X. Sean Wang. Transpcfg: Transferring the grammars from short passwords to guess long passwords effectively. *IEEE Trans. Inf. Forensics Secur.*, 16:451–465, 2021.
[31] Markus Jakobsson and Mayank Dhiman. The benefits of understanding passwords. In Patrick Traynor, editor, *7th USENIX Workshop on Hot Topics in Security, HotSec'12, Bellevue, WA, USA, August 7, 2012*. USENIX Association, 2012.
[32] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. Acoust. Speech Signal Process.*, 35(3):400–401, 1987.
[33] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio López. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 523–537. IEEE Computer Society, 2012.
[34] Johannes Kiesel, Benno Stein, and Stefan Lucks. A large-scale analysis of the mnemonic password advice. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society, 2017.
[35] Saranga Komanduri. *Modeling the Adversary to Evaluate Password Strength with Limited Samples*. PhD thesis, CMU, 2016.
[36] Yue Li, Haining Wang, and Kun Sun. A study of personal information in human-chosen passwords and its security implications. In *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*, pages 1–9. IEEE, 2016.
[37] Zhigong Li, Weili Han, and Wenyuan Xu. A large-scale empirical analysis of chinese web passwords. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 559–574, 2014.
[38] Enze Liu, Amanda Nakanishi, Maximilian Golla, David Cash, and Blase Ur. Reasoning analytically about password-cracking software. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 380–397. IEEE, 2019.
[39] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.
[40] Sanam Ghorbani Lyastani, Michael Schilling, Sascha Fahl, Michael Backes, and Sven Bugiel. Better managed than memorized? studying the impact of managers on password strength and reuse. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 203–220. USENIX Association, 2018.
[41] Sanam Ghorbani Lyastani, Michael Schilling, Michaela Neumayr, Michael Backes, and Sven Bugiel. Is FIDO2 the kingslayer of user authentication? A comparative usability study of FIDO2 passwordless authentication. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 268–285. IEEE, 2020.
[42] Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. A study of probabilistic password models. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 689–704. IEEE Computer Society, 2014.
[43] William Melicher, Blase Ur, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Fast, lean, and accurate: Modeling password guessability using neural networks. In *2017 USENIX Annual Technical Conference, USENIX ATC 2017, Santa Clara, CA, USA, July 12-14, 2017.*, 2017.
[44] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63(2):81–97, March 1956.
[45] Arvind Narayanan and Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, pages 364–372, New York, NY, USA, 2005. ACM.
[46] Miranda Lee Pao. An empirical examination of lotka's law. *J. Am. Soc. Inf. Sci.*, 37(1):26–33, 1986.
[47] Dario Pasquini, Ankit Gangwal, Giuseppe Ateniese, Massimo Bernaschi, and Mauro Conti. Improving password guessing via representation learning. *IACR Cryptol. ePrint Arch.*, 2019:1188, 2019.
[48] Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online, July 2020. Association for Computational Linguistics.
[49] qntm. L33t transformations, 2005. https://qntm.org/l33t.
[50] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018.
[51] Ashwini Rao, Birendra Jha, and Gananand Kini. Effect of grammar on security of long passwords. In Elisa Bertino, Ravi S. Sandhu, Lujo Bauer, and Jaehong Park, editors, *Third ACM Conference on Data and Application Security and Privacy, CODASPY'13, San Antonio, TX, USA, February 18-20, 2013*, pages 317–324. ACM, 2013.
[52] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.
[53] Richard Shay, Saranga Komanduri, Adam L. Durity, Phillip (Seyoung) Huh, Michelle L. Mazurek, Sean M. Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Can long passwords be secure and usable? In Matt Jones, Philippe A. Palanque, Albrecht Schmidt, and Tovi Grossman, editors, *CHI*

Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014, pages 2927–2936. ACM, 2014.

[54] Jaryn Shen, Timothy T. Yuen, Kim-Kwang Raymond Choo, and Qingkai Zeng. AMOGAP: defending against man-in-the-middle and offline guessing attacks on passwords. In Julian Jang-Jaccard and Fuchun Guo, editors, *Information Security and Privacy - 24th Australasian Conference, ACISP 2019, Christchurch, New Zealand, July 3-5, 2019, Proceedings*, volume 11547 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2019.

[55] Joshua Tan, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Practical recommendations for stronger, more usable passwords combining minimum-strength, minimum-length, and blocklist requirements. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1407–1426. ACM, 2020.

[56] Blase Ur, Saranga Kom, Richard Shay, Stephanos Matsumoto, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Michelle L. Mazurek, and Timothy Vidas. Poster: The art of password creation, 2013.

[57] Blase Ur, Fumiko Noma, Jonathan Bees, Sean M. Segreti, Richard Shay, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. "i added '!' at the end to make it secure": Observing password creation in the lab. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 123–140, Ottawa, July 2015. USENIX Association.

[58] Blase Ur, Sean M. Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L. Mazurek, William Melicher, and Richard Shay. Measuring real-world accuracies and biases in modeling password guessability. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, pages 463–481, 2015.

[59] Rafael Veras, Christopher Collins, and Julie Thorpe. On semantic patterns of passwords and their security impact. In *21st Annual Network and Distributed System Security Symposium (NDSS 2014), San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014.

[60] Ding Wang, Haibo Cheng, Ping Wang, Xinyi Huang, and Gaopeng Jian. Zipf's law in passwords. *IEEE Trans. Information Forensics and Security*, 12(11):2776–2791, 2017.

[61] Ding Wang, Haibo Cheng, Ping Wang, Jeff Yan, and Xinyi Huang. A security analysis of honeywords. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[62] Ding Wang, Debiao He, Haibo Cheng, and Ping Wang. fuzzypsm: A new password strength meter using fuzzy probabilistic context-free grammars. In *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2016, Toulouse, France, June 28 - July 1, 2016*, pages 595–606. IEEE Computer Society, 2016.

[63] Ding Wang, Ping Wang, Debiao He, and Yuan Tian. Birthday, name and bifacial-security: Understanding passwords of chinese web users. In Nadia Heninger and Patrick Traynor, editors, *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 1537–1555. USENIX Association, 2019.

[64] Ding Wang, Zijian Zhang, Ping Wang, Jeff Yan, and Xinyi Huang. Targeted online password guessing: An underestimated threat. In *Proceedings of ACM CCS, Vienna, Austria, October 24-28, 2016*, pages 1242–1254, 2016.

[65] Matt Weir. The version of 4.1 for pcfg models, 2019. https://github.com/lakiw/pcfg_cracker.

[66] Matt Weir, Sudhir Aggarwal, Breno de Medeiros, and Bill Glodek. Password cracking using probabilistic context-free grammars. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, pages 391–405, Washington, DC, USA, 2009. IEEE Computer Society.

[67] Daniel Lowe Wheeler. zxcvbn: Low-budget password strength estimation. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 157–173, Austin, TX, 2016. USENIX Association.

[68] Ming Xu and Weili Han. An explainable password strength meter addon via textual pattern recognition. *Security and Communication Networks*, 2019:5184643:1–5184643:10, 2019.

[69] Jeff Jianxin Yan, Alan F. Blackwell, Ross J. Anderson, and Alasdair Grant. Password memorability and security: empirical results. *IEEE Security Privacy*, 2(5):25–31, 2004.

[70] Leah Zhang-Kennedy, Sonia Chiasson, and Robert Biddle. Password advice shouldn't be boring: Visualizing password guessing attacks. In *2013 APWG eCrime Researchers Summit*, pages 1–11, 2013.

[71] Moshe Zviran and William J. Haga. Password security: An empirical study. *J. Manag. Inf. Syst.*, 15(4):161–186, 1999.

A APPENDIX

A.1 Experiments for passwords with length longer than 32 characters

For completeness, we also show the results for those extremely long passwords (i.e., longer than 32 characters) and observe their cracking rates separately. We note that the number of these extremely long passwords with more than 32 characters is much smaller. Concretely, in `Chinese` datasets, `CSDN`, `Youku` and `178` contain 0, 141, and 1 of these extremely long passwords, respectively. While in `English` datasets, `Rockyou`, `Neopets` and `Cit0day` contain 1,534, 208 and 7,585 of these extremely long passwords, respectively. We thus make this evaluation in `English` datasets. Following the experimental scenarios described in Section 4.1.1, we train `Rockyou` and combine `Neopets` and `Cit0day` as an evaluation set, and show the results in Figure 11. Particularly, CKL_Backoff achieves a higher efficiency after 10^{10} guesses, while CKL_FLA performs similarly with the *character-level* FLA model. CKL_PCFG cracks the same passwords with competitors, as evidenced in Min_auto strategies. Although our *chunk-level* models do not improve the guessing efficiency significantly, this may be because that these extremely long passwords are generally strong enough to resist guessing attacks, base on the less than 3% cracking rates. This phenomenon also confirms that the length is still a dominate factor for password security [30, 33, 53].

Further, we observe that these cracked extremely long passwords are homogeneous, indicating their vulnerability. Besides, to better understand their properties, we observe and count that most (around 67.91%) of these longer passwords are Email addresses, possibly because that users tend to adopt their Email addresses' formats as their secure passwords for easy memorization. As for the average number of chunks, we count that these longer passwords contain more chunks (i.e., 17.23) among four evaluation sets averagely.

A.2 Detailed changes after each *merge operation* of *PwdSegment*

We note there may be three changes in the size of the chunk vocabulary after each *merge* operation:

- *+1*: adding the newly merged chunk to the generated vocabularies and the original two chunks are still preserved. (The two original chunks do not always appear together at the same time.)
- *+0*: adding the newly merged chunk, and one of the original two chunks is preserved, the other is eliminated. (One of the original chunks appears exactly as another original chunk appears.)
- *-1*: adding the newly merged chunk, and both of the original chunks disappeared. (The two original chunks always appears together at the same time.)

Actually, the size of chunk vocabularies usually increases and then decreases trivially as the number of *merge operations* increases.

A.3 Detailed techniques of identifying memory patterns used in this paper

The following is the specific techniques to identify four memory patterns.

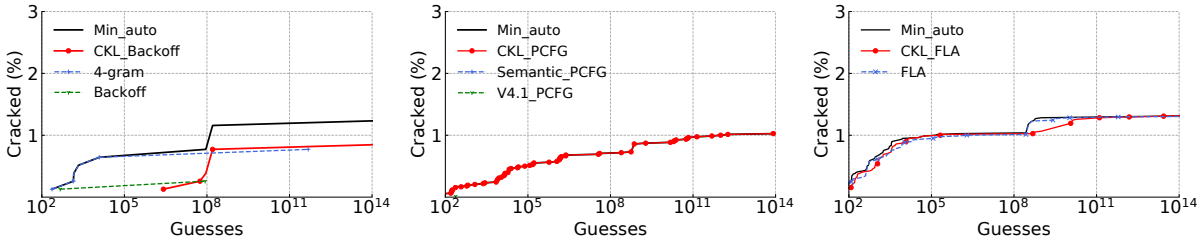


Figure 11: Guessing performance of longer passwords with more than 32 characters across three *chunk-level* guessing models.

Figure 12: Top-150 chunks in a shorter resulting chunk vocabulary (i.e., *avg_len* of 1.8).

• *Leet:* To identify leet patterns, we replace these digits or symbols with the characters based on the rulelist [49] and our additional rules [11], then compare the replaced passwords with the password base dataset (we refer to the *Rockyou* and *CSDN* as the base dataset for English and Chinese passwords, respectively). For example, we restore *p@ssw0rd* to *password* and then once the *password* is included in Rockyou, we label the *p@ssw0rd* as leet pattern password.
• *Syllable*: We refer to the English word-build affixes [11] and the Chinese pinyins together as syllables, where the dictionary includes exactly 528 (400 with pinyins, and 128 with English affixes) items.
• *Keyboard:* A substring where each character is adjacent to the next one in a standard English keyboard, e.g., the same row "zxcvbnm", the same column "1qaz2wsx" or zigzag "1qazse4rfv";
• *Date:* We refer to the consecutive four, six and eight digits with certain patterns as date according to the literature [37].

A.4 Top-150 shorter chunks compared with words

We also supplement the top-150 chunks in a sh orter resulting chunk vocabulary (i.e., the *avg_len* of 1.8) in Figure 12. Similarly, these four memory patterns (i.e., leet, syllable, keyboard and date) generally exist in these short chunks. Particularly, the number of **syllable** chunks (e.g, ing, wang) is much more significant in short chunks than that in long chunks (e.g., *avg_len* of 4.5). For syllable chunks, most of these common chunks are in the form of English word suffixes (e.g., ing) and Chinese pinyins (e.g., han). Especially, the surnames (e.g., wang or eng) are common in consisting chunks of Chinese datasets. The "love" theme plays an important role in Chinese culture, reflecting in top-150 chunks of 520 or 1314. These findings are generally consistent with that in long chunks.

[11] Our added rules: 8->ate; 2->too,to; 4->for,fore.