Improved Wordpcfg for Passwords with Maximum Probability Segmentation

Li, Wenting; Yang, Jiahong ; Cheng, Haibo ; Wang, Ping; Liang, Kaitai

DOI
10.1109/ICASSP49357.2023.10096535

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

Citation (APA)
Li, W., Yang, J., Cheng, H., Wang, P., & Liang, K. (2023). Improved Wordpcfg for Passwords with Maximum Probability Segmentation. In *Proceedings of the ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* IEEE. https://doi.org/10.1109/ICASSP49357.2023.10096535

ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10096535

IMPROVED WORDPCFG FOR PASSWORDS WITH MAXIMUM PROBABILITY SEGMENTATION

Wenting Li[1] *Jiahong Yang*[1] *Haibo Cheng*[1] *Ping Wang*[1] *Kaitai Liang*[2]

[1] Peking University [2] Delft University of Technology
Email: {wentingli,jiahongyang,hbcheng,pwang}@pku.edu.cn Kaitai.Liang@tudelft.nl

ABSTRACT

Modeling password distributions is a fundamental problem in password security, benefiting the research and applications on password guessing, password strength meters, honey password vaults, etc. As one of the best segment-based password models, WordPCFG has been proposed to capture individual semantic segments (called words) in passwords. However, we find WordPCFG does not address well the ambiguity of password segmentation by maximum matching, leading to the unreasonable segmentation of many password and further the inaccuracy of modeling password distributions. To address the ambiguity, we improve WordPCFG by maximum probability segmentation with A*-like pruning algorithm. The experimental results show that the improved WordPCFG cracks 99.26%–99.95% passwords, with nearly 5.67%–18.01% improvement.

Index Terms— Password, probabilistic context-free grammar, maximum probability segmentation.

1. INTRODUCTION

Password, as one of the primary authentication methods, suffers from guessing attacks and attracts many attentions on its strength [1, 2]. An adversary should first guess the passwords with high probabilities and then the low ones. This leads to the fundamental problem of password security: "what is the password distribution?" Several models have been proposed to model password distributions, such as PCFG [3], Markov [4], FLA [5], etc., and are further used in password applications, including password strength meters [6, 7], password leakage detection [8, 9], and honey password vaults [10, 11].

Segment-based models, also known as PCFG-based models, may not be able to provide the best accuracy but the best explainability on how users generate passwords, guiding users to choose safer passwords. We mainly deal with this type of model in this work. The models use probabilistic context-free grammars (PCFGs) to capture password generation as the concatenation of several segments.

Weir et al. [3] designed the first PCFG for passwords. Their model only parses passwords based on character types, for example, "password123!" is treated as the concatenation of "password" (L_8), "123" (D_3), and "!" (S_1). Rafael et al. [12] improved PCFG by extracting English words from passwords with NLP techniques and English dictionaries. Chatterjee et al. [13] further used more dictionaries (e.g., including keyboard patterns and dates). Cheng et al. [14] proposed *WordPCFG* with bootstrap word-extraction techniques to extract individual semantic segments (called *words*) of passwords, which are much different from natural English words.

1.1. Our Contributions

We find that WordPCFG makes the model become ambiguous by introducing word segments, e.g., "password" can be segmented as "password" or "pass/word". WordPCFG simply uses maximum matching to segment passwords, leading to unreasonable segmentation of many passwords, e.g., "villayouth" is segmented as "vil/layout/h" instead of "villa/youth". Meanwhile, the word dictionary of WordPCFG extracted from a password corpus contains a large number of low-frequency words, which makes the problem even worse.

We improve WordPCFG by figuring out the ambiguity in a more reasonable approach — maximum probability segmentation. More concretely, we parse a password according to the partition with the highest probability. We say that there are two challenges to implement the improvement: 1) searching the segmentation space costs a lot of time; 2) in the training phase, the segmentation algorithm needs the probabilities but they are unknown at this time. To tackle the first challenge, we use an A*-like pruning algorithm to reduce search space. For the second one, we design a two-phase approach: 1) train an original WordPCFG with maximum matching; and further 2) train our improved WordPCFG with maximum probability segmentation, where the probabilities are calculated by the original one. Meanwhile, we cut out the low-frequency words to reduce unreasonable words. The experimental results show that on the task of password guessing, our improved WordPCFG can crack 5.67%–18.01% more passwords than the original PCFGs, demonstrating the improved performance on modeling password distributions.

The first two authors contribute equally. Haibo Cheng and Ping Wang are the corresponding authors.

2. BACKGROUNDS

2.1. Password Models

The security of passwords depends on password distribution. Several password models have been proposed in different research topics on password security, e.g., password guessing [1, 2], password leakage detection [8, 9], password vaults [10, 11]. Less precisely, password models can be divided into two categories: segment-based and char-based. The former (e.g., [3, 14]) divides a password into several individual segments, and assigns its probability to the product of the structure probability and the probability of each segment. The latter (e.g., [4, 5]) treats each char in a password as a state, directly models the transition probability of each char based on the previous chars, and assigns the probability to the product of the transition probability of each char.

Segment-based models naturally capture the process of humans generating passwords, providing explainability on how users generate passwords and bringing advice on safer password generation. This work deals with this type of password models. And they usually leverage PCFGs to complete the formalization, which is thus also called PCFG-based models or PCFGs.

2.2. Existing PCFG-based models

Formally, a probabilistic context-free grammar (PCFG) is defined as a five-tuple $G = (S, N, \Sigma, R, p)$: S is the start symbol; N is the non-terminal set including S; Σ is the terminal set; R is the production rule set where the rules are in the form of $X \to str$, $X \in N, str \in (N \cup \Sigma)^*$; p is the probability density function on R, where $\sum_{str \in (N \cup \Sigma)^*} p(X \to str) = 1$ for any $X \in N$. Naturally, PCFG defines the probability distribution on the corresponding language ($\subseteq \Sigma^*$): the probability of a string in the language is its probability of being generated by the production rules.

Weir et al. [3] introduced PCFG to password models. Their model (denoted as W-PCFG in this paper), divides a password according to character types (letter, number, or other). For example, "password123!" is generated by four rules: $S \leftarrow L_8 D_3 S_1$, $L_8 \leftarrow$ password, $D_3 \leftarrow$ 123, and $S_1 \leftarrow$!. W-PCFG does not consider semantics in passwords. Further, Rafael et al. [12] extracted English words in passwords and constructed R-PCFG with NLP techniques. Chatterjee et al. [13] improved R-PCFG with more dictionaries of dates, keyboard-pattern and etc., yielding C-PCFG. Cheng et al. [14] used a bootstrap method to extract individual semantic segments from passwords, getting many segments different from natural language words (e.g., Jordon23 is marked as W_8 instead of $L_6 D_2$, 4ever is marked as W_5 instead of $D_1 L_4$), and further proposed WordPCFG.

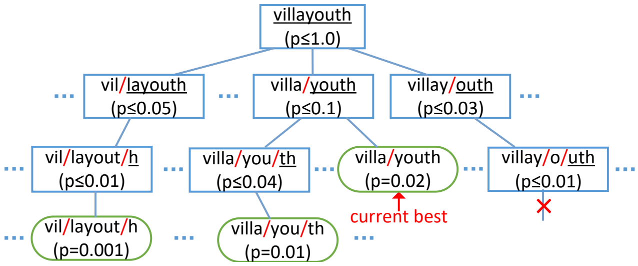


Fig. 1: The search tree for "villayouth".

3. OUR IMPROVED WORDPCFG

We first show the ambiguity of password segmentation in WordPCFG, and then use maximum probability segmentation to address the ambiguity, yielding a better WordPCFG.

3.1. Ambiguity in WordPCFG

By introducing word segments, WordPCFG becomes ambiguous, compared with unambiguous W-PCFG. More precisely, in WordPCFG, a password may be segmented into different schemes, e.g., "Jordon23" can be segmented as "Jordon23" (W_8) or "Jordon/23" ($W_6 D_2$). Cheng et al. [14] uses maximum matching to address the ambiguity, i.e., segmenting the longest word in the password first and doing the same recursively for the remaining two ends. However, the segmentation algorithm is local greedy, leading to unreasonable segmentation of many password, e.g., "villayouth" is segmented as "vil/layout/h" instead of "villa/youth" (more examples are given in Table 1).

3.2. Maximum probability segmentation

To handle the ambiguity, we propose the maximum probability segmentation. The basic idea is to find the segmentation scheme with the maximum probability, which usually leads to the most reasonable segmentation. However, the space of all possible segmentation schemes is exponentially sized, thus directly traversing the space is not feasible. We propose an A*-like pruning algorithm to narrow the search space.

The main idea of pruning is to 1) estimate the upper bound of the probabilities of all possible schemes (leaves) in a subspace (subtree) and 2) prune the subtree if the upper bound is less than the probability of the current best.

We give an example "villayouth" to illustrate our algorithm. As in Fig. 1, the current best is "villa/youth" with the probability 0.02, meanwhile in the subtree of "villay/o/uth", none scheme has a probability greater than 0.01. Thus the subtree can be pruned. Here, $x/\underline{Y}$ represents the space $\{x/y \mid y$ is a segmentation scheme for $Y\}$. The upper bound is estimated by $\Pr(L_6 \leftarrow \text{villay}) \Pr(L_1 \leftarrow \text{o}) \max_{T \in L_6 L_1 *} \Pr(S \leftarrow T)$, where $L_6 L_1 *$ represents the space of strings prefixed by $L_6 L_1$.

The calculation of $\max_{T \in L_6 L_1*} \Pr(S \leftarrow T)$ can be done within $O(1)$ time complexity by a pre-computed prefix tree recording the maximum probability for each prefix. We also construct another prefix tree of all the extracted words to speed up the decision on if a substring is a valid word. To further speed up the search, we first expand the subtree $x/\underline{Y}$ with the longest x. Since a reasonable segmentation scheme generally does not consist of too many segments, this technique helps the algorithm get better schemes earlier in the run and thus trigger the pruning more frequently.

3.3. Training

In the training phrase, the maximum probability segmentation algorithm needs the probabilities of each rules, but the probabilities are counted by the segmentation. To address the problem, we propose a bootstrap training with two phases. In phase 1, we first extract words from the password dataset and use maximum matching to train the original WordPCFG. During this phase, some unreasonable schemes may occur in the segmentation, but most of the words still get a reasonable estimation of their frequency. In phase 2, we parse the passwords with our maximum probability segmentation algorithm and use this new segmentation result to train a new model. In this phase, the probabilities provided to the segmentation algorithm are from the original WordPCFG. Phase 2 is necessary because it naturally reduces the influence of unreasonable segmentation schemes from phase 1. At last, we cut out the low-frequency words (less than 2) to further reduce the probability of overfitting. We note that word filtering could benefit the speed of the segmentation algorithm as fewer words are handled.

4. EXPERIMENTS AND RESULTS

4.1. Password Datasets

To present a fair comparison with the original WordPCFG, we use the same password datasets in the experiments, including Rockyou, 000Webhost, Clixsense, CSDN, Dodonew, and Duowan. These datasets have been leaked and public for years. As previous studies [1, 2, 3, 4, 15] in password security, we only use passwords in the datasets and do not leverage other information, e.g., personal information. This usage is ethical, since users will not be further harmed but may get better password protection from the research.

The six datasets are from various services, including social networks, web hosting, online surveys, developer community and games. The first three are for English-speaking users, and the others are for Chinese users. The datasets contain a total of 77.7 million accounts. For detailed statistics, please refer to [14]. The diversity of the datasets benefits the soundness and reliability of the experimental results.

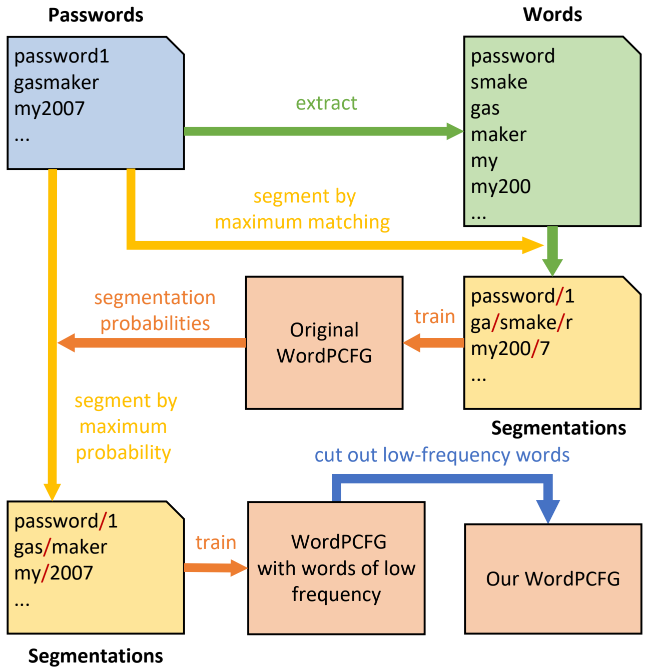


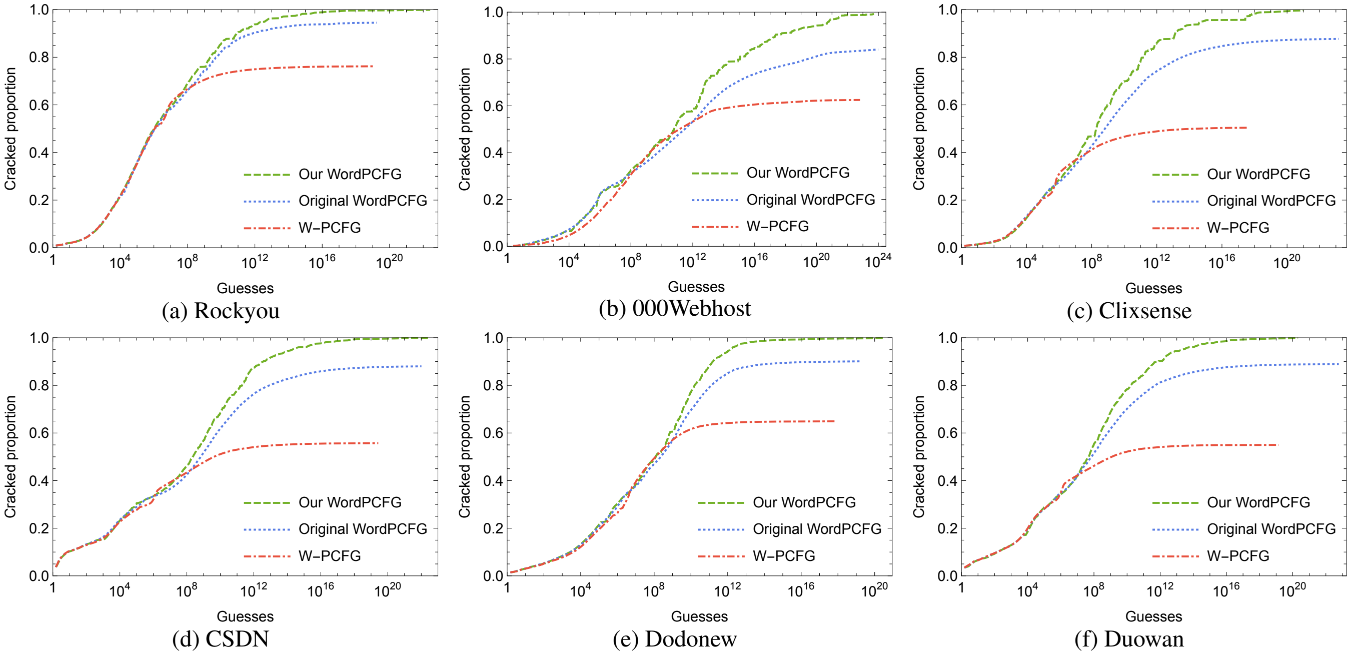
Fig. 2: The training process for Our WordPCFG

Table 1: Different segmentation examples: our WordPCFG and original WordPCFG

Password	Our WordPCFG	Original WordPCFG
weblogic1982	web/logic/1982	weblog/ic1982
my2007	my/2007	my200/7
ironearth680	iron/earth/680	ir/onearth/680
CarolineUS715	Carol/ine/US/715	Ca/roline/US7/15
196691	1966/91	19669/1
chemicalstudent	chemical/student	chemicals/tuden/t
beans+1bugs	beans/+1/bugs	beans/+/1bug/s
clixbbaa11	clix/bbaa/11	clixb/baa1/1
gasmaker	gas/maker	ga/smake/r
19mira635	19/mira/635	19mi/ra63/5

4.2. Performance on Segmentation

We first compare the performance between maximum probability segmentation and maximum matching segmentation. Table 1 presents some typical examples that differ between these two algorithms. Take the password "ironearth680" as an example, it can be naturally split into three segments "iron/earth/680", which is the same with the maximum probability segmentation. As "onearth" is also an extracted word, maximum matching will first separate it from the password, then the part "ir" is a dead end. This unreasonable segmentation will severely underestimate the probability of "ironearth680". Maximum probability segmentation avoids this problem by considering all segments comprehensively and thus significantly improves the quality of segmentation.



(a) Rockyou (b) 000Webhost (c) Clixsense

(d) CSDN (e) Dodonew (f) Duowan

Fig. 3: The performance of WordPCFGs for password guessing attack

Table 2: Cracked proportions under the given guess numbers

Dataset	Model	10^4	10^{10}	10^{18}	Final
Rockyou	Our WordPCFG	21.60%	**85.74%**	**99.62%**	**99.95%**
	Original WordPCFG	**22.14%**	82.30%	94.42%	94.59%
	W-PCFG	21.67%	72.85%	76.18%	76.23%
000Webhost	Our WordPCFG	6.42%	**45.50%**	**91.13%**	**99.26%**
	Original WordPCFG	**7.23%**	41.52%	77.46%	84.11%
	W-PCFG	4.72%	44.68%	61.43%	62.53%
Clixsense	Our WordPCFG	12.46%	**69.92%**	**98.38%**	**99.86%**
	Original WordPCFG	**13.23%**	60.58%	86.45%	87.74%
	W-PCFG	12.80%	46.62%	50.42%	50.45%
CSDN	Our WordPCFG	23.25%	**68.50%**	**99.33%**	**99.93%**
	Original WordPCFG	**23.55%**	61.83%	87.26%	88.07%
	W-PCFG	22.69%	51.12%	55.68%	55.72%
Dodonew	Our WordPCFG	**13.29%**	**77.15%**	**99.68%**	**99.87%**
	Original WordPCFG	13.03%	69.62%	90.01%	90.20%
	W-PCFG	12.14%	61.61%	64.91%	64.94%
Duowan	Our WordPCFG	18.50%	**78.36%**	**99.48%**	**99.88%**
	Original WordPCFG	19.70%	70.19%	88.44%	88.91%
	W-PCFG	**19.98%**	52.17%	55.00%	55.01%

4.3. Performance on Password Guessing

Password guessing attack is a practical approach to reflect the model accuracy and provides an objective metric to make comparisons among different models. We compare our improved WordPCFG with the original WordPCFG [14] and W-PCFG [3]. As in [14], each dataset is randomly divided into two halves for training and testing, respectively.

The guessing results are shown in Fig. 3 and Table 2. For small guessing numbers ($< 10^8$), the numbers of cracked passwords under the three models are basically the same; but for larger guessing numbers ($> 10^{12}$), our improved WordPCFG performs significantly better than the original WordPCFG, proving that maximum probability segmentation and two-phase training are effective to WordPCFG. It is worth noticing that we successfully push PCFG-based models to a nearly 100% cracking rate, which is the best performance in the research line.

5. CONCLUSION

We improve WordPCFG by maximum probability segmentation to handle the ambiguity in password segmentation. The new WordPCFG is able to parse passwords more appropriately and perform better on password guessing. With better accuracy on modeling password distributions, our model may bring benefit to the research line of password security and real-world password applications.

Acknowledgment

This research is supported by National Natural Science Foundation of China under Grant 62072010, 62202012, China Postdoctoral Science Foundation under Grant 2021M700215, 2022T150013, National Key R&D Program of China under Grant 2020YFB1805400, European Union's Horizon 2020 research and innovation programme under grant agreement No. 952697 (ASSURED), No. 101021727 (IRIS), and No. 101070052 (TANGO), and High-performance Computing Platform of Peking University.

6. REFERENCES

[1] D. Pasquini, A. Gangwal, G. Ateniese, M. Bernaschi, and M. Conti, "Improving password guessing via representation learning," in *Proc. IEEE S&P 2021*, pp. 1382–1399.

[2] M. Xu, C. Wang, J. Yu, J. Zhang, K. Zhang, and W. Han, "Chunk-level password guessing: Towards modeling refined password composition representations," in *Proc. ACM CCS 2021*, pp. 5–20.

[3] M. Weir, S. Aggarwal, B. D. Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *Proc. IEEE S&P 2009*, pp. 391–405.

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

[5] L. Bauer, W. Melicher, B. Ur, S. M. Segreti, and L. F. Cranor, "Fast, lean, and accurate: Modeling password guessability using neural networks," in *Proc. USENIX Security 2016*, pp. 175–191.

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

[7] M. Golla and M. Dürmuth, "On the accuracy of password strength meters," in *Proc. ACM CCS 2018*, pp. 1567–1582.

[8] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *Proc. ACM CCS 2013*, pp. 145–160.

[9] D. Wang, Y. Zou, Q. Dong, Y. Song, and X. Huang, "How to attack and generate honeywords," in *Proc. IEEE S&P 2022*, pp. 966–983.

[10] H. Cheng, Z. Zheng, W. Li, P. Wang, and C.-H. Chu, "Probability model transforming encoders against encoding attacks," in *Proc. USENIX Security 2019*, pp. 1573–1590.

[11] H. Cheng, W. Li, P. Wang, C.-H. Chu, and K. Liang, "Incrementally updateable honey password vaults," in *Proc. USENIX Security 2021*, pp. 857–874.

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

[13] R. Chatterjee, J. Bonneau, A. Juels, and T. Ristenpart, "Cracking-resistant password vaults using natural language encoders," in *Proc. IEEE S&P 2015*, pp. 481–498.

[14] H. Cheng, W. Li, P. Wang, and K. Liang, "Improved probabilistic context-free grammars for passwords using word extraction," in *Proc. IEEE ICASSP 2021*, pp. 2690–2694.

[15] B. Pal, T. Daniel, R. Chatterjee, and T. Ristenpart, "Beyond credential stuffing: Password similarity models using neural networks," in *Proc. IEEE S&P 2019*, pp. 814–831.