The Zhuyin Password Cracker

Lien-Bee Huang
National Taiwan University
Taipei, Taiwan
b04505021@ntu.edu.tw

Jason Liang
National Taiwan University
Taipei, Taiwan
r08922008@ntu.edu.tw

Jr-Wei Huang
National Taiwan University of Science and Technology
Taiwan, Taiwan
m10815007@mail.ntust.edu.tw

Ying-Chiao Chen
National Taiwan University
Taipei, Taiwan
b07901184@ntu.edu.tw

Abstract

With different native languages, people would set their passwords in different patterns. Moreover, people type in a non-English layout would create a seemingly secure password. However, with knowledge about such layout, it might not be secure as expected. Zhuyin, also known as bopomofo, is a phonetic notation system that can notate the pronunciation of Chinese characters and it uses its layout apart from English layout. Therefore, we would like to explore how many passwords are using the Zhuyin pattern and how they set their password. In this paper, we analyze the passwords from Taiwan, a country mainly use Zhuyin. According to our research, a lot of people set their passwords in similar patterns and structures. In terms of this observation, we enhance Probabilistic Context-Free Grammar (PCFG) by the Zhuyin feature; Eventually, in evaluation, we can mutate Zhuyin patterns successfully and hit much more real passwords than the other tools.

CCS Concepts:* • Security and privacy → *Cryptanalysis and other attacks.

Keywords: Zhuyin, password cracker

ACM Reference Format:
Lien-Bee Huang, Jason Liang, Jr-Wei Huang, and Ying-Chiao Chen. 2020. The Zhuyin Password Cracker. In *CNS '20: Cryptography and Network Security Final Project, June 30, 2020, Taipei, TW*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Though many new authentication approaches come up, text password is still the most widely using one. Because it is easy to deploy, it would even not be replaced in decades.

To evaluate the strength of a password, people use password estimators to help. Even there are lots of password strength estimators, text passwords are not strong enough. Passwords created by non-English arrangements would make password strength estimators overestimate the strength of those passwords. When people with relative background knowledge, those "strong" passwords might be cracked easily. Zhuyin is a phonetic notation system that can easily create a seemingly secure password.

For example, a leakage detection service "HIBP"[2] shows that a seemingly secure password "ji32k7au4a83" leaked hundreds of times. In fact, in Zhuyin layout, "ji32k7au4a83" is "ㄨㄛˇㄉㄜ˙ㄇㄧˋㄇㄚ" and it can be converted into "我的密碼" as Chinese characters, which means "my passwords".

To explore how many passwords are creating in Zhuyin, what kind of patterns they used, and whether they are strong enough, we collect leaked publicly available password DBs, which contains 1740153 passwords in ".tw" domain (Section 3); then we extract and analyze the Zhuyin patterns by our Zhuyin detection algorithm (Section 4).

To effectively guess the Zhuyin passwords, we adopt the PCFG based guessing algorithm and improve existing algorithm with Zhuyin feature (Section 5). With our improvement, we reach the most hit and unique Zhuyin hit among existing tools (Section 6).

In this project, we make the following contributions:

1. **Zhuyin Detection Algorithm**: Zhuyin passwords are combinations of digits, alphabets, and symbols. To detect Zhuyin passwords, we detect whether there are valid patterns and examine whether such a password a reasonable Zhuhyin password.
2. **Common patterns of Zhuyin passwords**: According to our findings, the Zhuyin passwords in some similar patterns. We categorize those patterns and alert people not to use passwords in such patterns.

3. **Improvement of Zhuyin password mutations**: As a mutation-based guessing algorithm, PCFG can generate several passwords to crack. With the Zhuyin feature, the PCFG cracker can hit many more Zhuyin passwords which not appear in the Training dataset than the other tools.

2 Problem Definition

We are interested in three research questions: (1) What patterns are Taiwanese like to use in passwords. (2) How many Taiwanese use Zhuyin-related passwords (3) Given that an attacker is aware of the Zhuyin password distribution, can he mutate different Zhuyin-related password. It is necessary to address these questions to let both security engineers and Chinese users know the strength of Zhuyin-related passwords. We assume that the attacker has a Taiwanese password training-set, so they can learn the Zhuyin distribution and guess more efficiently.

In our attack model, the attacker has leak files containing many user passwords. The goal for the attacker is to crack as many passwords as possible from another protected database. The attacker does not know any user password in that database. The only thing he needs to do is to learn from the leak passwords files he has and use it to attack the database. The attacker is allowed to use any mutation or learning method to derive guesses. However, he has no permission to attack the database to get the password directly.

3 Analysis

3.1 Dataset

To analyze, we collect publicly available leaked passwords. Because the majority of Zhuyin user is Taiwanese, we extract 1.7 million passwords under ".tw" domain. For the following attacking steps, we use another leaked password database with 162 thousand passwords. The training set would be used to analyze and provided PCFG to build up its probabilistic priority lists.

3.2 Distribution

With our Zhuyin detection algorithm (Section 3), there are around 5% passwords is Zhuyin form. For those Zhuyin passwords, we simply divide them into different types in terms of their grammar structure. According to Table 1, we can realize that users who create passwords in the Zhuyin form prefer to set them with pure Zhuyin instead of a combination with alphabets, digits, or symbols.

3.3 Common Patterns

By survey the common Zhuyin passwords, we found there are lots of similar patterns. There we categorize them into the following parts.

- **Showing love**: Surprisingly, The most common category is to show love. For example, "ji394su3" (means "I love you") appears 2948 times in this dataset, and "ji3cp394su3" (means "I love you very much") appears 39 times. And also, many passwords mean "I love *someone*".
- **Name**: In this category, people like to create their passwords from names, including real names, nicknames, their idol's names, or even some of their favorite character's names. Due to the custom in Taiwan, almost all of the real names are 3 connected Chinese.
- **Profanity**: As our expectation, profanity is one of the major categories. For instance, there are 25 passwords to be "e04su3su;6" ("fuck your mother").
- **Others**: Apart from the above categories, some people like to create their password in a short sentence. The structure of those password sentences would be like "Subjective + Verb + Objective" or "Subjective + be + Complement".

To sum up, those Zhuyin passwords have similar patterns. We can try some guessing algorithms to hit those passwords. In our work, we adopt the PCFG based guessing algorithm (Section 5).

4 Zhuyin Detection

First of all, we need to know whether a Zhuyin combination is valid, so we refer to this website [1]. Now when we get a password, we read it character by character from the beginning. Whenever we found a character that is actually a Zhuyin tone on the keyboard that contains Zhuyin, we try to find the longest valid Zhuyin combination. Take "hahaji3g45j" for example, the first tone appears to be "3" then we start from "aji3" (a valid Zhuyin combination will have length at most 4), discard the first character until we find a valid Zhuyin combination, and it appears to be "ji3" ("我"). After looping over the password, we will translate it into ("haha我是5j").

To further reduce false-positive and detect more possible Zhuyin, we introduce the following methods.

1. *Detect space bar*: For some passwords, the space bar, which may represents the first tone in Zhuyin, does not exist (like the above example "hahaji3g45j"). To resolve this problem, we try to insert the space bar at the end of each English segment after detecting possible Zhuyin. For example, "hahaji3g45j" will first become "haha我是5j", then after inserting space bar, the password will become "haha 我是5j ", then we will try to detect Zhuyin again and remove needless space bar, so the final answer will become "haha我是豬"
2. *Filter unlikely password*: We use three rules to filter out unlikely Zhuyin passwords.
   a. Since most Zhuyin tone will be typed as number (except for space), we won't try to detect Zhuyin in a password that can be partitioned into the English part plus number part.

Table 1. The number of each grammer with Zhuyin

Grammar	Pure Zhuyin	Zhuyin+Alphabet	Zhuyin+Digit	Zhuyin+Alphabet+Digit
Number	17,615	274	129	152
Ratio	96.8%	1.5%	0.7%	0.8%

b. We filtered out the password that after Zhuyin detection, the Zhuyin parts contain only digital numbers. This is because there are only a few meaningful Zhuyin combinations using only digital numbers, to reduce false positive, we choose to discard these few Zhuyin combination.
c. We've mentioned that we use a dictionary to check if a Zhuyin combination is valid. However there exists some rare Zhuyin combination (e.g. ㄎㄚˋ 髂), we filter them out manually.

Despite the methods proposed above, there are still too many false-positives, so we introduce the following more aggressive rules. We will choose to use some of them according to whether we want more false positive or less.

1. *Full Chinese*: This is the most aggressive rule. We will only consider a password as a Zhuyin password when after detecting Zhuyin, all the Chinese words (i.e. Zhuyin combinations) are connected together.
2. *Three Connected Chinese*: This one is less aggressive than *Full Chinese.* We will only consider a password as a Zhuyin password when after detecting Zhuyin, there exist at least three Chinese words that are connected together.
3. *Two Connected Chinese*: The only difference between this and *Three Connected Chinese* is that it requires only two connected Chinese words.

5 PCFG

It seems that we can only use the brute-force method to guess the password if we have no information about the users. However, that's not the case. The distribution of the human-made password is not uniform. Therefore, we can use this property to optimize the guessing process.

In our project, we choose PCFG(Probabilistic Context-Free Grammar) as our model for password cracking, which is one of the state-of-the-art methods to guess the password efficiently and effectively.

5.1 The PCFG Rules

In the PCFG model, a string can be regarded as a combination of several segments. For example, "Jason123" consists of two individual parts: "Jason" and "123". Each segment can be classified into a class. "Jason" belongs to class "A", which means the alphabet, where "123" belongs to class "D", which means digit. Therefore, the structure of "Jason123" is "A5D3". The number follows the class means how many consecutive characters can be identified as that class. The structure "A5D3" is called "grammar" in the PCFG model.

In the PCFG tool [3] we use, it can be separated into two parts: the trainer and the guesser.

5.2 The PCFG trainer

The PCFG trainer is to process the passwords in the given training data set and then create the grammar for the PCFG guesser to do password cracking. It has the following steps:

1. Read passwords into the trainer and check the encoding of all passwords.
2. For any given password, it will use many detection methods to detect patterns and then parse a password into several segments. For example, if a password is "Jason123ji394su3", its structure is "Z5D3Z3".
3. After parsing all passwords in the training data set, it will calculate the probabilities of each segment and save them into the Rules folder.

5.3 The PCFG Guesser

After the training process completes, the PCFG guesser can generate many guesses stemmed from the rules. Let's take "A4D4", for example, to illustrate the process of the guesser.

Based on the result[5], the probability of "A4D4" is 0.039, "love" is 0.0835, and "1234" is 0.04, respectively. Then the string "love1234" has a probability of 0.039 * 0.0835 * 0.04, where three events are independent. The guesser calculates many patterns and then insert them into a priority queue. The guesser will pick the item with the highest probability as its next output. The guesser keeps doing this guessing process until all combinations in the Rules are generated or when we press the stop button.

5.4 Improvement

In our work, we integrated the Zhuyin detector mentioned above into the PCFG tool. Our goal is to crack more Zhuyin than the original PCFG. In the PCFG trainer, we use *Three Connected Chinese* instead of *Two Connected Chinese* in order to lower the false positive rate to make sure our training set is not contaminated by some garbages. On the PCFG guesser side, we use *Two Connected Chinese* to check whether the guesses contain Zhuyin or not. This comes from the nature of PCFG. The passwords that the PCFG guesser generates may contain Zhuyin substring less than 2. On the other hand, the passwords generated by the PCFG guesser has a lower

Table 2. Comparison between different tools

Tool	Attempt	Hit	Zhuyin	Unique
Dictionary	1.74 M	34876	757	0
John	8.6 M	37273	782	27
hashcat	8.6 M	17332	94	12
Original PCFG	8.6 M	45159	2	2
Zhuyin PCFG	8.6 M	47285	558	79

Table 3. PCFG with different converages

Coverage	Attempt	Hit	Zhuyin	Unique
All PCFG(1)	100 M	64332	570	95
Mixed(0.8)	100 M	67377	655	148
Mixed(0.6)	100 M	68278	729	202
Mixed(0.4)	100 M	63868	596	151
Mixed(0.2)	100 M	61976	703	189

false-positive rate. Therefore, using *Two Connected Chinese* for detection in guesses is reasonable and will not affect our final result.

In order to measure the accuracy of PCFG with the Zhuyin feature in Taiwanese passwords, we construct two experiments scheme 1 and scheme 2.

On scheme 1, we create a Taiwanese password data-set and then leverage four different password cracking tools to crack the testing-set in limited guessing time. On the data-set part, to create a Taiwanese password training-set, we filtered .tw email address from 41G password leak list and clean up some error passwords like hash. Our testing target including john, hashcat, original PCFG, and Zhuyin PCFG. We limit each guessing time in 8.5 million to make sure every tool is equal in the testing.

On scheme 2, we use another feature provided by the PCFG tool[3]. The coverage means what percentage do you expect the target password's base words can be found in the training set. If the coverage = 0.6, then the model is a hybrid of the PCFG model and Markov model with the ratio 6:4. The Markov model is another approach for password cracking, it can be seen as an optimized version of the brute-force method but using some techniques to improve its performance. By changing the ratio between the PCFG model and the Markov model, we try to find the coverage that leads to the best outcome.

6 Evaluation

Our results show in Table 2 and 3. Each row uses a different method we mentioned above to train and guess. The data in column 2 is the password guess and column 3 means how many guesses hit the testing-set passwords. By leveraging the Zhuyin detection, we divided column 3 into non-Zhuyin passwords and Zhuzin passwords(column 4). Column 5 shows how many Zhuzin passwords we hit don't exist in the training set.

Table 2. shows our experiment results using 5 different ways to crack the passwords. Except for the dictionary attack without the mutation (the Dictionary row in Table 2.), we guess the password for 8.6M times since the default dictionary mutation rules of John The Ripper can only generate that much of guessing. Although the brute-force methods(john) can get better Zhuyin hits, its total hits are much lower than the PCFG model. If we focus on the PCFG model only, we can see that our improved Zhuyin PCFG has higher performance in all aspects, especially in unique Zhuyin.

Table 3. shows the difference between each coverage. The limitation of PCFG is the reason why the mixed PCFG, especially when coverage = 0.6, has a better performance than the pure PCFG. The pure PCFG can only generate guesses that can be combined using the segments parsed in the training set. For instance, if "kiwi" does not exist in the training set, then any string containing "kiwi" will never be cracked using the pure PCFG tool. Nonetheless, if we add part of the Markov model into the PCFG model, we can expand the domain of the guessing set, which can help us to crack those passwords that are not shown in the training set and then increased the password hits.

The interesting things in table 3 are that as the proportion of the Markov model increases, the password hits first reach a maximum point and then decreases. But the Zhuyin hit when coverage is 0.2 will bounce back in contrast to coverage is 0.4. It can be seen as the extension of the brute-force method as shown in table 2(john). Moreover, since Mixed(0.2) is still benefited from the PCFG model, the unique Zhuyin hit rate will be higher than Mixed(0.4). However, the total hits will slightly reduce, which is affected by the brute-force method as expected.

7 Related Work

In this section, we briefly review previous research about password cracking and security.

Li et al. [7] create a large-scale analysis of passwords leaked from five Chinese websites and compared with the RockYou and yahoo password data-set. They observed that Chinese users tend to use a special pattern in the password. For Instance, Chinese users prefer to use the format YYYYMMDD and YYMMDD, but not in English data-set. In order to improve the efficiency of cracking Chinese passwords, they generate different data-set, rule-set, and dictionaries to test which combination has the best result.

Wang et al. [10] analysis of 73.1 million real-world Chinese web passwords including six Chinese websites. And measured the password data-sets in terms of length distribution, popular structures, letter distribution, and frequency. To evaluate the strength of the Chinese web password, they use two password crackers (PCFG-based and Markov-based).

Furthermore, they found a weakness in PCFG-based crackers and improve it to crack more passwords.

There are several password cracker tools that support a dictionary attack along with some mutations [4] [9], they can utilize CPU, or even GPU to do highly parallel cracking with their mutation rules. Even though they can produce great results most of the time, there aren't rules or methods related to Zhuyin employed by them.

PCFG[11] is one of the state-of-the-art cracking models.Its approach was a considerable addition to the existing techniques in password cracking. The process can be divided into training and testing. On the training part, passwords were separated into different data types and count their associated probabilities. Use the probabilities to decide the order.

Another approach for password cracking is using the Markov Model[8]. In the Markov model, it uses Markov chains to determine which characters are to be selected to form its next guess. It calculates the probabilities of each character based on the occurrence of each character in the data set, which is also called frequency analysis. We can imagine that if we pick the most possible character, it will be more likely to hit the true password. In most cases, choosing the top 50% characters are able to cover over 90% possible guesses.

Also, some researches tried to categorize some common mutation rules. Based on the rules, their tool can mutate the inputted dictionary into the mutated one and successfully guess more passwords. "The Tangled Web of Password Reuse"[6] used many mutation rules such as capitalizations, Reversals, Leet-speak, substring movement, and Subword modification to generate new passwords. Meanwhile, zxcvbn[12] adapts a similar mutation approach: leet-speak, capitalizations, and reversals. Such mutations rules would be another direction to improve the guessing algorithms.

8 Future Work

Even though we get significant improvement in Zhuyin password cracking, we found many possible ways to improve our tools and leave them as future work.

1. *Chinese name*: We found that many people using their Chinese names as passwords, and since PCFG utilizes n-grams for password guessing, these names will definitely ruin the n-grams mode. It is possible to delete these names from the password database, and somehow find the most common Chinese names (maybe in other leaked databases). When a PCFG needs to fill in a "Z3" or "Z2", we can try to fill in these names.
2. *NLP models with DL*: PCFG uses Markov models for its NLP (natural language processing), however we may employ deep learning to train an NLP model with Zhuyin on our huge database.
3. *More Zhuyin detecter*: As mentioned in **Zhuyin Detection** part, our Zhuyin detection model can only detect *Full Chinese*, *Two Connected Chinese* and *Three Connected Chinese*, we may loss some Zhuyin passwords such as "3us493ij" (an inverse of "ji394su3", which is effectively 我愛你)
4. *Stronger Zhuyin detector*: Since our Zhuyin detector will try to find the longest possible Zhuyin whenever a tone is met, passwords like "loveji394su3" will be detected as lov果愛你, however, it should be love我愛你. obviously this will mess up the PCFG trainer. We have not solved this problem yet, however, a possible solution is to search detected Chinese sentence online (e.g. Google search) and see how many results are found. If the results' numbers are fairly high, we may believe that our detection is correct.

9 Conclusion

We propose a method for parsing out Zhuyin parts in a password with help of our analytic results of common Zhuyin patterns. After implemented this method into PCFG, we get significant improvement on password cracking targeting Zhuyin passwords. Specifically, our Zhuyin PCFG find 77 Zhuyin passwords that are not in our training set, while the original PCFG can only find 2. Since our Zhuyin detection method can be implemented into most tools that analyze words structure in passwords, e.g. n-grams, our results show that using Zhuyin password may not be as secure as it might seems. However our cracker still cannot handle well at mangled password (english and Zhuyin mixed), people can try to use mixed english and Zhuyin phrases to mitigate our attack tool.

References

[1] 2013. 標準國語的讀音變化統計. https://blog.xuite.net/fg_wang/twblog/106485279-%E6%A8%99%E6%BA%96%E5%9C%8B%E8%AA%9E%E7%9A%84%E8%AE%80%E9%9F%B3%E8%AE%8A%E5%8C%96%E7%B5%B1%E8%A8%88.
[2] 2013. Have i been pwned. https://haveibeenpwned.com/.
[3] 2013. PCFG Cracker. https://github.com/lakiw/pcfg_cracker.
[4] 2015. hashcat. https://hashcat.net/hashcat/.
[5] 2020. CNS final demo. https://docs.google.com/presentation/d/1oJF5kDDyzvuRrKSe6uHARuRiBOkTP4MOxhqsbrn-RO8/edit#slide=id.g8ad9b0c77f_5_10.
[6] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and Xiaofeng Wang. 2014. The Tangled Web of Password Reuse.
[7] Zhigong Li, Weili Han, and Wenyuan Xu. 2014. A Large-Scale Empirical Analysis of Chinese Web Passwords. In *23rd USENIX Security Symposium (USENIX Security 14)*. 559–574.
[8] Arvind Narayanan, Vitaly Shmatikov, and Wenyuan Xu. 2005. Fast dictionary attacks on passwords using time-space tradeoff. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*. 364–372.
[9] solardiz. 2002. John The Ripper. https://github.com/magnumripper/JohnTheRipper.
[10] Ding Wang, Ping Wang, Debiao He, and Yuan Tian. 2019. Birthday, Name and Bifacial-security: Understanding Passwords of Chinese Web Users. In *28th USENIX Security Symposium (USENIX Security 19)*. 1537–1555.

[11] Matt Weir, Sudhir Aggarwal, Breno De Medeiros, and Bill Glodek. 2009. Password cracking using probabilistic context-free grammars. In *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 391–405.

[12] Daniel Lowe Wheeler. 2016. zxcvbn: Low-Budget Password Strength Estimation. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 157–173. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/wheeler