applied sciences

Article

Dynamic Markov Model: Password Guessing Using Probability Adjustment Method

Xiaozhou Guo [1,2], Yi Liu [1,2], Kaijun Tan [1,2], Wenyu Mao [1,2], Min Jin [1,2] and Huaxiang Lu [1,2,3,4,*]

[1] Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China; xiaozhouguo@foxmail.com (X.G.); liuyi@semi.ac.cn (Y.L.); tankaijun@semi.ac.cn (K.T.); maowenyu@semi.ac.cn (W.M.); jinmin08@semi.ac.cn (M.J.)
[2] University of Chinese Academy of Sciences, Beijing 100089, China
[3] CAS Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Beijing 200031, China
[4] Semiconductor Neural Network Intelligent Perception and Computing Technology Beijing Key Lab, Beijing 100083, China
[*] Correspondence: luhx@semi.ac.cn; Tel.: +86-010-82304210

Abstract: In password guessing, the Markov model is still widely used due to its simple structure and fast inference speed. However, the Markov model based on random sampling to generate passwords has the problem of a high repetition rate, which leads to a low cover rate. The model based on enumeration has a lower cover rate for high-probability passwords, and it is a deterministic algorithm that always generates the same passwords in the same order, making it vulnerable to attack. We design a dynamic distribution mechanism based on the random sampling method. This mechanism enables the probability distribution of passwords to be dynamically adjusted and tend toward uniform distribution strictly during the generation process. We apply the dynamic distribution mechanism to the Markov model and propose a dynamic Markov model. Through comparative experiments on the RockYou dataset, we set the optimal adjustment degree α. Compared with the Markov model without the dynamic distribution mechanism, the dynamic Markov model reduced the repetition rate from 75.88% to 66.50% and increased the cover rate from 37.65% to 43.49%. In addition, the dynamic Markov model had the highest cover rate for high-probability passwords. Finally, the model avoided the lack of a deterministic algorithm, and when it was run five times, it reached almost the same cover rate as OMEN.

Keywords: information security; password guessing; Markov model; machine learning; dynamic mechanism; probability adjustment

check for updates

Citation: Guo, X.; Liu, Y.; Tan, K.; Mao, W.; Jin, M.; Lu, H. Dynamic Markov Model: Password Guessing Using Probability Adjustment Method. *Appl. Sci.* **2021**, *11*, 4607. https://doi.org/10.3390/app11104607

Academic Editor: Junggab Son

Received: 25 March 2021
Accepted: 11 May 2021
Published: 18 May 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the development of information, more and more services need security protection, and identity authentication has gradually become a primary means to protect users' information. Password-based authentication was created with the advent of mainframes, and has been widely used in mainframe access control since the 1960s. Now, passwords have become one of the most critical means of preventing users from being attacked in the Internet world [1]. Passwords are not only easy for users to understand and memorize, but also relatively simple for programmers to deploy at low cost, so they have been generally recognized and adopted by academia and industry [2]. Password authentication will still be a crucial method in the future [3].

However, on the one hand, users often need to manage dozens or hundreds of password accounts, and the numbers are growing. Additionally, the requirements for setting passwords on various websites are often very different [4]. On the other hand, users' energy for dealing with information security affairs is pretty limited, and this will not be greatly improved over time [5]. This fundamental contradiction leads to potentially vulnerable behaviors, such as using weak passwords with low information entropy, reusing the same password on multiple websites, and recording passwords on paper [6]. All of these behaviors can bring out vulnerable factors including simple numbers, familiar words, too short length, the inclusion of personal information, and so forth. Consequently, the vulnerable factors make attacking passwords possible [7].

Password attack technology is mainly used for recovering passwords [8] and measuring their strength [9]. In the virtual environment, passwords are often set for files to protect specific data. If users forget or lose their passwords for encrypted files, it is necessary to recover them. In addition, sometimes the security department will need to recover illegal documents [10], which will also include operating system password recovery, mailbox password recovery, encrypted file password recovery [11], and so forth. The primary purpose of evaluating password strength is to analyze the ability of a given plaintext password to resist brute force cracking [12]. In general, some passwords can be cracked with some techniques or tools, so in the field of password strength measurement, the number of times or time spent cracking is taken as the evaluation index.

The password attack method utilizing the password generative model is an offline attack [13] that does not need to interact with the server. The password can be saved in either plaintext or hash value, and the number of password guessing is usually significant. It is a kind of trawling attacking method [14], which does not use any personal user information (such as birthday, email, etc.) during the attack. Accompanied by the development of machine learning technology, various password generative models have been proposed [15–17]. Attackers first generate a large number of passwords using a generative model, then try to crack the system using the generated passwords one by one. Most of the early password generative models relied on heuristic rules set by experts, such as replacing lowercase letters in passwords with uppercase letters, changing words, and so forth [18]. These pure rule-based models have poor universality, and their effect is not satisfactory. Recently, models based on machine learning have become mainstream, such as the Markov [19], PCFG [20], RNN(LSTM) [21], and GAN [22] models. They explicitly or implicitly model the probability of a password with mathematical theory foundation support, and perform better in practice. Compared with the other three models, the Markov model, based on statistical machine learning, has certain advantages in training and inference speed, because it only needs to count and randomly sample instead of performing many neuronal operations [23]. At the same time, it maintains almost the same level of generated password quality because its assumptions accord with human language habits. The Markov model is still widely used because of its superior comprehensive performance [24].

Nevertheless, our observation finds that the Markov model suffers from a high repetition rate, and this problem becomes more severe as the number of passwords the model generates increases. It is obvious that a high repetition rate will also lead to an insufficient cover rate [25]. The enumeration method is different from the random sampling method, which is a deterministic algorithm. It usually has a worse cover rate of high-probability passwords, and always generates the same passwords in the same order when generating them [26], which makes it easy to attack. To solve the above problems, we introduce a dynamic distribution mechanism into the random sampling method. Then, we add a dynamic distribution mechanism to the Markov model, and eventually build a dynamic Markov model.

The structure of this paper is as follows. In Section 2, we briefly introduce other password cracking methods, including the brute force cracking and dictionary methods. We introduce the password generative models mainly include the statistical learning models (such as Markov and PCFG model) and the deep learning models (such as RNN and GAN model). In Section 3, we first analyze the shortcomings of the standard Markov model based on random sampling and OMEN based on enumeration. For the above problems, we introduce the dynamic distribution mechanism and prove its mathematical basis. Finally, we apply the dynamic distribution mechanism to the Markov model and propose a dynamic Markov model. In Section 4, we carry out the experiment, and compare the performance of several related models from multiple perspectives. In Section 5, we summarize and analyze the proposed model based on theoretical analysis and experiments. The last section presents our conclusion.

2. Related Work

In order to improve the success rate of password cracking, the patterns of passwords have been studied intensely. Through the analysis of most users' passwords [27,28], it is found that in addition to choosing some words as passwords, users often simply change words to meet the requirements of a website password setting strategy, and about 1 in 10 people choose to use the top 10 passwords. By mining several leaked public data sets, Wang concluded that passwords, like human languages, all satisfy Zipf distribution [29], that is, the frequency of the password decreases by polynomial, and high-frequency and low-frequency passwords occupy important parts of the whole password set [30]. The phenomenon that most Chinese passwords contain numbers, and more than a quarter of them are only composed of numbers, was found by [31]. English passwords prefer English letters, and most passwords are composed of letters and numbers according to an analysis of the character structure of passwords [32]. The study of the length of passwords indicates that more than 90% of passwords are between 6 and 11 characters, but for websites with high importance, the password length is usually greater than 11 characters [33]. Additionally, name, name ID, birthday, username, email prefix, ID number, phone number, and even love information may be used [34].

Generally speaking, password cracking methods consist of brute force [35] and dictionary [6] cracking. In the brute force cracking method, with multi-thread, distributed, and GPU assistance [36], an attacker can test potential passwords through exhaustive algorithms in a relatively simple password space. However, as the maximum password length and character set size increase, the size of the password space will grow exponentially, making it difficult to traverse and search all passwords under limited computing resources [37]. The dictionary cracking method generates a dictionary consisting of a variety of passwords and guesses using the dictionary. Furthermore, some rules are adopted to promote diversity. John the Ripper [38] designs word transformation rules that simulate the scene when the user creates a password. Hashcat [39] modifies, cuts and expands words, and adds some human preferred rules. Analogously, it has inefficient performance when the string space becomes vast. Rather than depending on traverse searching, which ignores the password difference or sets heuristic rules, password generative models try to generate high-quality passwords by machine learning theory.

2.1. Markov Model

Narayanan first introduced the Markov model in the password guessing task [19]. The password possesses local relevance, which means each character only has a relationship with the previous n characters. The order of generating passwords from left to right conforms to human habits, and the concise assumption is reasonable in natural language processing. Based on the Markov model, Tansey expanded the number of layer to n and equipped each layer with different weights, where n represents the expected length of the password [40]. Each node in the multi-layer Markov model not only can switch to itself, but also save related information of front characters. Due to the supplemented probability distribution, the model improves the quality of password generation. OMEN [26], proposed by Durmuth, can generate passwords in descending order of frequency. Then, passwords with high probability appear early, which contributes to increasing the cracking speed. Wang improved the Markov model for targeted attacks [41]. Specifically, it treats username, email prefix, name, birthday, phone number, and identification number as equal characters. The targeted Markov model makes full use of the user's personal information and shows excellent performance. The Markov model, as a classic algorithm for natural language processing, also shows good effects with passwords. Additionally, considering the speed perspective, the Markov model surpasses the other password generative models [42]. In a word, the Markov model is still popular for its overall performance.

2.2. Other Password Generative Models

Other password generative models use distinct principles and can achieve a great cracking effect. The PCFG model [20] cuts the password according to the type of character (including letter (l), number (n), and special symbol (s)). It counts all generated structures from the training dataset and calculates the probability information during training, eventually uses a queue to generate passwords. Houshmand adopted keyboard rules in FCFG [43], and could crack extra passwords that obey these rules. Veras conducted deep semantic mining of the letter part, which particularly increased the cracking ability for long passwords [44]. Li proposed a personal-PCFG model that equally dealt with username, email prefix, and so forth. The reinforced model has better performance especially in target attacks [45]. With the success of deep learning, neural networks that have more capacity and stronger representation ability have attracted attention. Sutskever generated text using recurrent neural networks, showing that the RNN model has the ability to handle text sequences [46]. Melicher took passwords as an extension of human language [21], and first tried to generate passwords in the RNN framework. The RNN generated one character at each time step and used the generated character as the input for the next time step until the terminator appeared or the maximum length was reached. In other studies, RNN [47] was replaced with LSTM [48], promoting the effect of capturing the long-range dependence of characters. Teng devised a PG-RNN that increased the number of neurons, and got competitive results on multiple datasets [34]. The generative adversarial network [15] is currently the best among the generative models, especially in computer vision. PassGAN was introduced in [22] to generate passwords. Then, PassGAN was enhanced [49] to directly learn the probability distribution of the password encoding matrix. After training with the Wasserstein loss function [50,51], it just needs to provide the generator with random noise sampled in Gaussian distribution, and then the generator outputs the coding matrix of the password. Nam used the relative GAN to improve the loss function, and the performance of GAN in password guessing was greatly improved by multi-source training [52]. In addition, the generator was improved in [53] with recurrent neural networks, which could acquire better generation effect.

Overall, first, compared with the Markov model, the relatively low speed is an apparent disadvantage for the other models [23]. In the training stage, the PCFG model needs to additionally count structure probability information. For the models based on neural networks, many weight parameters must be trained by the backpropagation algorithm [54]. In the generation stage, the PCFG model has to generate extra structure. The RNN and GAN models contain large scale multiplication operations and activation functions. These complicated operations inevitably slow down inference speed in contrast with random sampling [55]. Second, viewed from the perspective of quality, the PCFG model's straightforward splitting of strings deviates from the actual situation, and models based on neural networks have limit representation ability in passwords [56]. In addition, considering the problem of generating repeated passwords, models based on neural networks usually do not perform satisfactorily if they are not trained well; in particular, the GAN model frequently experiences mode collapse [57]. Statistical machine learning models such as the Markov model and PCFG models, also generate large-scale repeated passwords if the random sampling method is used. In summary, the Markov model has an advantage in terms of comprehensive performance, yet a high repetition rate would be one factor prohibiting the model from cracking efficiently.

3. Method

3.1. Standard Markov Model

It is impossible to accurately model the probability of a password directly. The Markov model makes a concise hypothesis for the expression of password probability, and then the password can be statistically modeled. In the Markov model, the probability distribution of characters in each state is only related to the characters in the front n state, where n is defined as the order of the model, as shown in Figure 1.

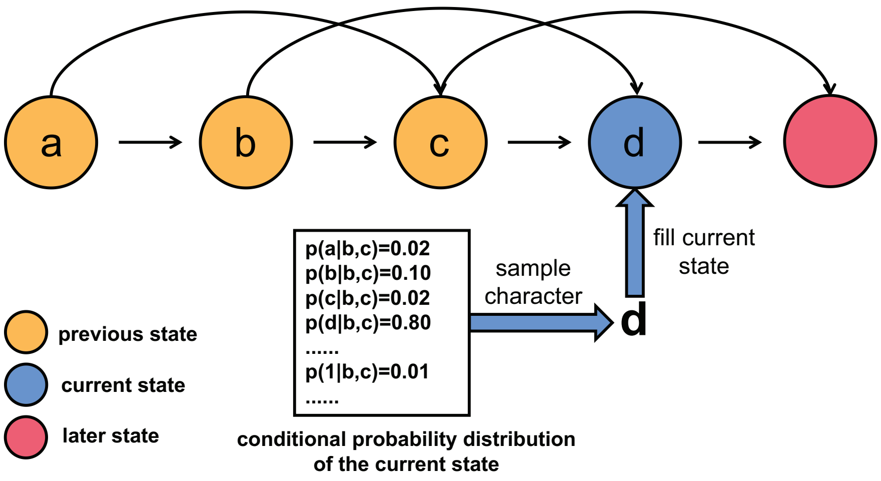

Figure 1. Markov model (order of model is 2).

Each character in the password can be combined with the previous n characters to form an n-gram fragment, which is denoted as $c_i|c_{i-1},\ldots,c_{i-n}$ where c_i is the character in the ith state, and $|$ is used to isolate the current and previous state characters, so the n-gram fragment represents the dependencies between the current and previous character. For example, a password $P : c_1c_2\ldots c_m$ containing m characters can be divided into m n-gram fragments: $c_1; c_2|c_1; \ldots; c_m|c_{m-1}, c_{m-2}, \ldots, c_{m-n}$. It should be noted that the first n n-gram fragments are not n order gram, which should strictly be 0-gram, 1-gram, n-gram. Here, we call them all n-gram fragments for convenience. Each n-gram fragment $c_i|c_{i-1},\ldots,c_{i-n}$ corresponds to a conditional probability $p(c_i|c_{i-1}, c_{i-2}, \ldots, c_{i-n})$, and these n-gram fragments are independent of each other, so the probability of password P is the product of m conditional probability values:

$$p(c_1c_2\ldots c_m) = \prod_{i=1}^{m} p(c_i|c_{i-1}, c_{i-2}, \ldots, c_{i-n}). \tag{1}$$

The Markov model should be trained well before generating passwords. In preparation, we have to collect a public password dataset containing quantities of real passwords as a training set. While training the Markov model, we first split each training password into multiple n-gram fragments, count the occurrence frequency of all n-gram fragments, and then collect n-gram fragments that have the same conditional character; eventually we perform the probability normalization operation to obtain the corresponding n-gram conditional probability distribution. The process is depicted in Figure 2. We define the n-gram table as the set of all n-gram conditional probability distributions. After the training process succeeds, the n-gram table stores the conditional probability distribution of all characters learned by the model, and the explicitly statistical modeling of password probability is completed. When we need to generate a password, we just continuously sample the characters in the n-gram table one by one. We denote this kind of Markov model that uses random sampling as the standard Markov model in the next.

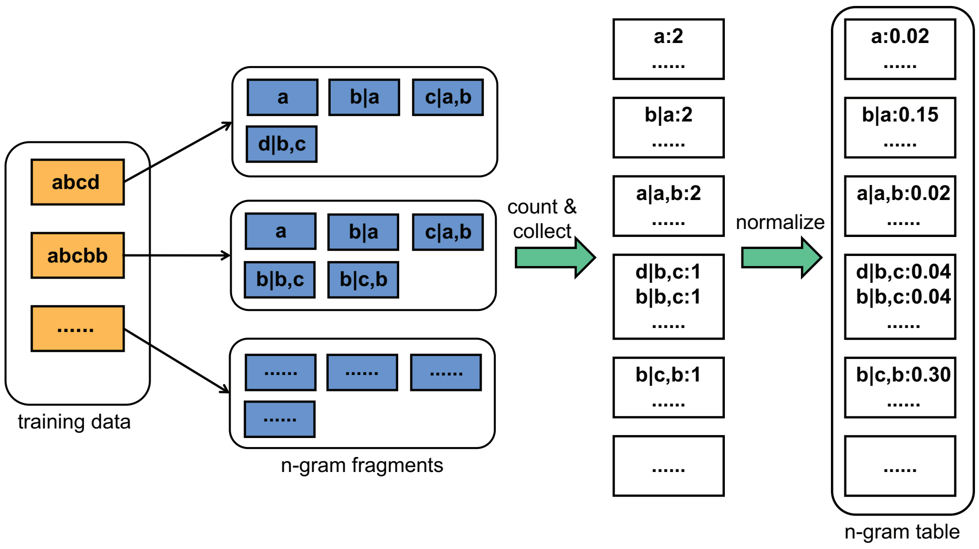

Figure 2. Markov model training process (order of model is 2).

In order to further analyze the password probability distribution, we have to make a clear definition of the support set and original probability distribution of the Markov model. We define the maximum length of the string as m_{MAX}, and the character set is $V = A, B, \ldots, Z, a, b, \ldots, z, 0, 1, \ldots, 9, +, *, \ldots, !$. For any string $C : c_1c_2 \ldots c_m$ in string space S_S, all satisfy both (1) $m \le m_{MAX}$, and (2) $c_i \in V, \forall i$. The Markov model explicitly fixes the probability of each string in string space S_S. For part of the strings $C \in S_S$, probability $P(C)$ can be obtained by querying the n-gram table, while for another part of strings $C^{'} \in S_S$, some n-gram fragments do not exist in the n-gram table, which means their probability is 0. We can form a set of all strings whose probability value is more than 0, and define this set support set S_M. Obviously, support set S_M is a subset of string space S_S. The sum of the probabilities of all strings in S_M is 1, where high probability means the string has a higher probability of being a password, and low probability means the string is not likely to be a password. Only a string with a probability more than 0 can be considered as a password. Any password the model may generate must belong to S_M, so the size of S_M means the cracking potential of the model. The training process not only enables the Markov model to learn support set S_M from string space S_S, but also assigns a corresponding probability to each password in S_M. We call this distribution the original distribution D_{ori}. It is apparent that original distribution D_{ori} is only determined by the training data set.

3.2. *Defects in Markov Model*

3.2.1. Problems in Standard Markov Model

Cracking passwords using the standard Markov model has the problem of generating large scale repeated passwords. To illustrate this problem, we utilized the RockYou [58] dataset to train the model. RockYou dataset is a large scale dataset that includes almost 21,000,000 real passwords, therefore it is widely used in password guessing tasks. We made the model generate 10^9 passwords, and we observed the result from two aspects: the number of repeated passwords and the repetition rate. The repetition rate expresses the proportion of repeated passwords in the generated passwords. The results (Table 1) show that there are only 2.41×10^8 unique passwords, and the other 7.59×10^8 passwords are all repeated passwords, which means the model wastes about 76% of computing resources. In addition, we find that there is a positive correlation between the number of generated passwords and the number and proportion of repeated passwords. More unique passwords are accompanied by more waste, which severely limits the cracking efficiency.

Repeated passwords not only constrain efficiency, but also restrict the cover rate. The cover rate indicates the hit ratio of generated passwords to the passwords in the test set. One reason is that repeated passwords take up many appearance opportunities within limit generation time, so the other passwords that are possible to hit have no chance to be generated. Another reason is that the essence of the standard Markov model is equal to random sampling in the original distribution D_{ori}. Due to the randomness of the sampling method, high-probability passwords will appear with high frequency, while low-probability passwords will not show up often or even disappear, and then passwords with low probability in the test set are hard to cover.

Table 1. Number of repeated passwords and repetition rate for standard Markov model.

Total Number	**10^5**	**10^6**	**10^7**	**10^8**	**10^9**
Repeated number	3.41×10^4	4.79×10^5	5.94×10^6	6.86×10^7	7.59×10^8
Repetition rate	34.12%	47.93%	59.37%	68.61%	75.89%

3.2.2. Problems in OMEN

Compared with the other Markov model, OMEN, which uses the enumeration method, gets the best cover rate result [26] but it does not perform best in any probability range. In our experiment, we first divided the RockYou test set into 10 regions according to the probability value (obtained by the Markov model, which is trained on the train dataset). OMEN was also trained in the training dataset and generated 10^9 passwords. We find that OMEN performs worse in the high probability range, as shown in Table 2. For example, in the range 10^{-5} to 10^{-6}, the standard Markov model hits 3000 more passwords.

Table 2. Cover numbers of three models for different probability of passwords.

Probability	**Total**	**Dynamic Markov**	**Standard Markov**	**OMEN**
10^{-2}∼10^{-3}	4	4	4	4
10^{-3}∼10^{-4}	243	243	243	243
10^{-4}∼10^{-5}	2701	2701	2701	2671
10^{-5}∼10^{-6}	19,827	19,827	19,826	19,456
10^{-6}∼10^{-7}	116,095	116,094	115,973	112,954
10^{-7}∼10^{-8}	284,911	284,897	271,945	268,945
10^{-8}∼10^{-9}	390,731	350,988	272,115	340,801
10^{-9}∼10^{-10}	410,065	122,463	95,250	287,433
10^{-10}∼10^{-11}	334,797	13,275	13,507	140,972
10^{-11}∼10^{-12}	231,826	980	1168	26,988
$<10^{-12}$	310,024	68	161	817

In addition, OMEN is a deterministic algorithm, while the standard Markov model is a stochastic algorithm. After the model finishes the training process and corresponding parameters are all fixed, OMEN always generates the same passwords with the same order each time, whereas the standard Markov generates different passwords each time. Thus, in some cases, determinacy could be used to prevent passwords from being cracked by OMEN as possible, while this is not possible in the standard Markov model. For instance, we generate T passwords as set S_d using OMEN, and randomly select a password P_{tar} that not in S_d. We also require that P_{tar} must have been modeled, that is, its probability is not zero. At this point, we can guarantee that OMEN will not hit P_{tar} within T generations, because it does not appear in the first T generations. However, it is possible for every password that has been modeled to emerge at any moment if the standard Markov model is applied, hence it has absolute advantages in this particular case.

3.3. Dynamic Distribution Mechanism

For all of the above problems, we propose the dynamic distribution mechanism during the generation process in the Markov model to reduce the repetition rate, so as to improve the cover rate. In establishing the dynamic distribution mechanism, we mainly consider four points, as follows:

First, the dynamic distribution mechanism is apply only in the random sampling method. Second, each time a password P_G is generated, the dynamic distribution mechanism is adopted. It prohibits P_G by reducing its probability value, thereby reducing the possibility of subsequent appearance of P_G. As the probability of the password in the Markov model is the product of multiple conditional probabilities, the passwords' probability expression has a complex interwoven relationship, then we can consider subtracting $p(P_G)$ by a small constant. Third, since the sum of the probability of all passwords in support set S_M is 1, prohibiting P_G would inevitably lead to a change in probability of other passwords in S_M. Here, we could increase the probability value of all passwords in S_M other than P_G, thus the model would focus more on obtaining other passwords in the next. Finally, it should be pointed out that for strings other than S_M, we think they have little meaning for cracking, and adjusting their probability would require a large amount of computing resources, so the dynamic distribution mechanism only changes the probability of passwords in support set S_M.

There are several advantages to the dynamic distribution mechanism:

1. It can decrease the repetition rate. Once a password is generated, it will be restricted, and the chance of being randomly sampled will be reduced. At the same time, the chance of other passwords being randomly sampled will be increased. The degree of constraint is related to the probability value. Considering the situation that the higher the password's probability, the larger its contribution to the repetition rate, so this mechanism can decrease the repetition rate accordingly. On the other hand, we find that the dynamic distribution mechanism pushes the probability distribution of passwords toward uniform distribution U_{uni} according to Theorem 1. The closer the probability distribution gets to the uniform distribution U_{uni}, the smaller the repetition rate.
2. It can improve the cover rate. Compared with simple random sampling, the dynamic distribution mechanism can help the model to adjust distribution constantly. This adjustment makes the model avoid useless attempts, but encourages it to try completely new passwords.
3. It can cover high-probability passwords better. In the random sampling method, the appearance frequency of passwords is relevant to their probability. The dynamic distribution mechanism designs a smooth and slow process which ensures that some passwords will keep a high probability value for a long time. At the same time, its effect of reducing repeated passwords expands the search range of high-probability passwords. Therefore, usually, high-probability passwords are almost all sampled if the number of generation operations is large enough. However, enumeration methods (such as OMEN) use local instead of global sorting, so they generate passwords in an approximate but not strict descending probability order. In fact, it is impossible to go through all the passwords and sort them strictly in descending probability order. Hence, usually some high-probability passwords are missed, which is also shown in the experiment in Table 2.

Theorem 1. *For any original distribution D_{ori}, randomly select a password P_i which satisfies $p_i^{ori} \times N \geq 1$, where N is total number of passwords in S_M, then reduce its probability by α , and increase the other passwords probability by $\alpha/(N-1)$, then the new passwords probability distribution D_{new} will be closer to the uniform distribution D_{uni}.*

Proof of Theorem 1. We use KL divergence $D_{KL}(D_{uni}||D_{ori})$ to measure the distance between the original and uniform distribution, which is:

$$D_{KL}(D_{uni}||D_{ori}) = -\lg N - \frac{1}{N}\sum_{i=1}^{N} \lg p_i^{ori}. \tag{2}$$

In the neighborhood of p^{ori}:$[p_1^{ori}, p_2^{ori}, \ldots, p_N^{ori}]$, we perform a one-order Taylor expansion about $D_{KL}(D_{uni}||D_{ori})$ which gets $f(p)$:

$$f(p) = -\lg N - \frac{1}{N}\sum_{i=1}^{N} \lg p_i^{ori} - \frac{1}{N}\sum_{i=1}^{N} \frac{1}{p_i^{ori}}(p_i - p_i^{ori}). \tag{3}$$

In D_{new}, the probability of P_i is $p_i^{ori} - \alpha$, while the other passwords probability is $p_j^{ori} + \alpha/(N-1)$ and $j \in 1, 2, \ldots, N\backslash i$. The approximate KL divergence $D_{KL}(D_{uni}||D_{new})$ is:

$$\begin{aligned} D_{KL}(D_{uni}||D_{new}) \approx -\lg N - \frac{1}{N}\sum_{i=1}^{N} \lg p_i^{ori} + \frac{1}{N}\frac{\alpha}{p_i^{ori}} \\ -\frac{1}{N}\sum_{j\in\{1,2,\ldots,N\}\backslash i} \frac{1}{p_j^{ori}}\frac{\alpha}{N-1}. \end{aligned} \tag{4}$$

The approximate difference of two KL divergence A is defined as:

$$A = D_{KL}(D_{uni}||D_{new}) - D_{KL}(D_{uni}||D_{ori}). \tag{5}$$

Therefore, A can be reduced to:

$$A = \frac{1}{N}\frac{\alpha}{p_i^{ori}} - \frac{1}{N}\sum_{j\in 1,2,\ldots,N\backslash i} \frac{1}{p_j^{ori}}\frac{\alpha}{N-1}. \tag{6}$$

According to the inequality:

$$\frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_n} \geq \frac{n^2}{x_1 + x_2 + \ldots + x_n}. \tag{7}$$

So,

$$A \leq \frac{1}{N}\frac{\alpha}{p_i^{ori}} - \frac{1}{N}\frac{\alpha}{N-1}\frac{(N-1)^2}{1-p_i^{ori}} \tag{8}$$

Obviously, if $p_i^{ori} \times N \geq 1$, we have $A \leq 0$, so $D_{KL}(D_{uni}||D_{new}) \leq D_{KL}(D_{uni}||D_{ori})$. □

3.4. Dynamic Markov Model

We employ the dynamic distribution mechanism in the standard Markov model and propose a dynamic Markov model. Given the way passwords are modeled by Markov chain, we have to make subtle adjustments in practice. There are three deviations compared with the ideal theory in Theorem 1.

When the model generates password $P_G : c_1c_2\ldots c_m$, it is impossible to verify whether $N \times p(P_G)$ is more than 1 or not, because N is unknown, therefore the model has to replace the verification method with another simplified method. As multiple n-gram fragments jointly determine the probability of the password, it is very complicated to deal with the probability of the whole password, so the dynamic Markov model executes operations at the n-gram level instead of the password level. So we judge whether the probability value of the n-gram fragment is greater than α. Moreover, the experimental results also prove that this simplification can effectively make the distribution of the password more uniform.

In order to reduce the probability of P_G, we can only adjust the conditional probability. If m conditional probability values are all adjusted, the model will consume too much calculation time, affecting the overall password generating speed. Simultaneously, a large number of passwords will be affected, and the probability distribution will change greatly, so it is difficult to guarantee the cover rate. For example, 123456 and 123456789 are both high-probability passwords; 123456 has been generated and 123456789 has not appeared. Adjusting 6 conditional probability values of password 123456 will greatly reduce the probability of 123456789, so randomly getting 123456789 becomes more difficult. Hence, the dynamic Markov model randomly selects one of m conditional probability to adjust each time. This approach of modifying the probability is matched with password modeling mode, and it limits the influence range of modifying as much as possible.

We need to increase the probability of passwords in S_M except for P_G, but it is impossible to traverse all of these passwords. Therefore, we increase the probability of some passwords which have the selected n-gram fragments, while the probability of other unrelated passwords reamins the same. The dynamic Markov model ensures that the changes of the password probability distribution just restrict on the support set S_M and do not propagate into the string space S_S. Although that only modifies a small part of the password, it avoids the time-consuming modification of all passwords every time.

Overall, because the number of passwords in the support set N is unknown and large, and Markov's modeling strategy for passwords makes them share n-gram fragments, the dynamic Markov model makes the above three modifications on the basis of comprehensive consideration of computational efficiency and practical difficulties. It successfully realizes the application of the dynamic distribution mechanism. It should be noted that there is a balancing effect in the dynamic distribution mechanism itself. For example, changing the probability distribution of the n-gram fragment $c|a,b$ would result in the over modification of some passwords (such as those that contain $c|a,b$; $d|a,b$; $e|a,b$; etc.), but would have no effect on other passwords (such as those that do not contain these above n-gram fragments). These two simultaneous effects would be partially offset during the multiple processes of random sampling to generate passwords.

Our dynamic distribution mechanism works as follows in practice: for example, first randomly select n-gram fragment $c_3|c_1,c_2$. For the discrete conditional probability distribution $p(c|c_1,c_2)$, reduce the value of $p(c_3|c_1,c_2)$ with a fixed value α and adjust it to $p(c_3|c_1,c_2)-\alpha$, increase the value of $p(\hat{c_3}|c_1,c_2)$ evenly, and keep the value of $p(\tilde{c_3}|c_1,c_2)$ always 0, where $\hat{c_3}$ represents the character not c_3 and appears in the conditional probability distribution $p(c|c_1,c_2)$, and $\tilde{c_3}$ is the other characters in character set V except c_3 and $\hat{c_3}$. In summary, when the model adjusts the conditional probability distribution $p(c|c_1,c_2)$, it automatically adjusts the probability distribution of other passwords in S_M. For passwords that contain n-gram fragments $c_3|c_1,c_2$, the probability decreases, and the probability of passwords containing fragments $\hat{c_3}|c_1,c_2$ increases, as shown in Figure 3. The change degree of the probability depends on the situation of including n-gram fragments. The complete password generating algorithm of dynamic Markov model is shown in Algorithm 1.

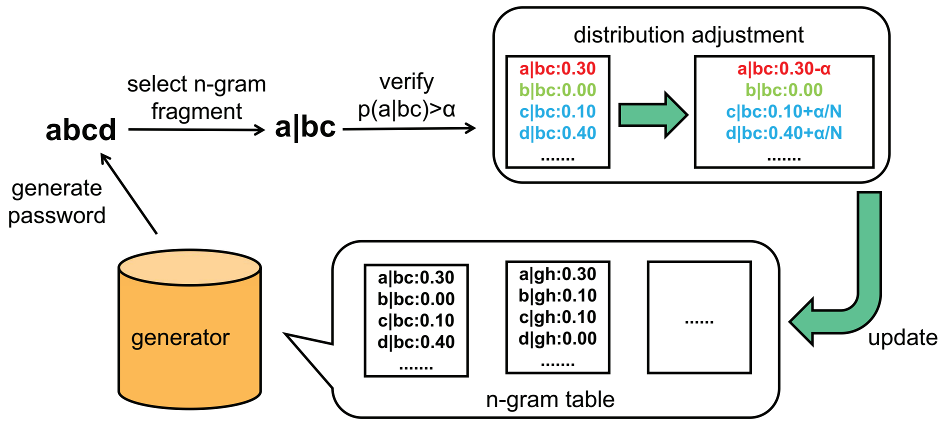

Figure 3. Generating passwords using dynamic Markov model.

Algorithm 1 dynamic Markov model

```
Require: number of generative passwords: $N$, max length of password: $m_{MAX}$, n-gram fragment: $c_3|c_1,c_2$, other characters except $c_3$ in distribution $p(c|c_1,c_2)$: $\hat{c_3}$, fixed value of reduced probability: $\alpha$.
for $i = 1$:$N$
  initial blank string $S$
  do
    generate random number $r \sim N(0,1)$
    generate a character $c$ by randomly sampling in n-gram table using $r$
    append $c$ to $S$
  until $c = \backslash n$ or $|S| = m_{MAX}$
  randomly choose an n-gram fragment (for example, $c_3|c_1,c_2$) in string $S$
  if $p(c_3|c_1,c_2) > \alpha$
    adjust $p(c_3|c_1,c_2)$ to $p(c_3|c_1,c_2) - \alpha$
    count total number of $\hat{c_3}$: $N_{dec}$
    for $\hat{c_3}$:
      adjust $p(\hat{c_3}|c_1,c_2)$ to $p(\hat{c_3}|c_1,c_2) + \alpha/N_{dec}$
    end for
  end if
end for
```

4. Experiments

To verify the performance of the dynamic Markov model, we chose to experiment on public dataset. The training of a password generative model requires a large and realistic data set, and the RockYou dataset satisfies both requirements. RockYou.com [58] stores passwords in plaintext form in the database, and the SQL vulnerability of the site led to a user password leak. Millions of passwords were stolen by intruders and spread on the network for many years; this dataset has been widely applied in academic research related to passwords. The RockYou dataset includes 21,315,685 passwords in total, among which 8,274,727 passwords are unique after deduplication. We randomly shuffled the data set initially, and took the first 80% of the data as the training set and the bottom 20% as the test set. The training set contained 17,052,548 passwords, with 6,909,743 unique passwords. The test set included 4,263,137 passwords, with 2,174,626 unique passwords .

In the experiment, the training set was used to train the password generative model, and both the cover rate and repetition rate were used to evaluate the performance. Assuming that the model generates N passwords and the number of unique passwords is n, the repetition rate RR is

$$RR(model) = 1 - \frac{n}{N}. \tag{9}$$

If the test set has T passwords and t unique passwords, then the cover rate CR is

$$CR(model) = \frac{n}{t} \tag{10}$$

Our experimental hardware environment is: Intel(R) Xeon(R) Silver 4116 CPU with 2.10 GHz main frequency, 12 cores and 128 GB memory. The software environment is: the Ubuntu 16.04.6 LTS operating system. The programming language is Python3.6.

4.1. Effect of Parameter α

There exists a vital parameter α that has a significant and decisive impact on the performance of the dynamic Markov model. This parameter represents the amplitude of each probability adjustment and directly controls the speed at which the probability distribution of passwords tends to be uniform. If α is too large, most of the n-gram fragments cannot be adjusted, because the probability value is less than α, and the cover rate is greatly reduced (for example, when α is 10^{-5}, the cover rate is less than 20%), making the model useless. On the contrary, if the value of the parameter is small, the effect difference between parameters is not apparent (for example, when α is set as 10^{-11} and 10^{-12}, the cover rate and repetition rate are almost the same), so we conducted experiments on parameter α from 10^{-5} to 10^{-11}, and tried a variety of different values, including $10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}$, and 10^{-11}. First, we used the training set to train the dynamic Markov models with different values of α, and we selected the model order n as 3. Next, we used these 7 password generative models to generate passwords. In order to evaluate the model comprehensively, we generated 10^9 passwords and calculate the CR and RR values in each scale. The experimental results are shown in Figures 4 and 5.

The experimental results prove that the scale of parameter α has a remarkable impact on the performance of the model. For the repetition rate, choosing a smaller α would get a larger RR value. When α is equal to 10^{-5}, RR is always below 0.5. As the parameter decreases, the gap between adjacent parameters becomes smaller. Both parameters 10^{-11} and 10^{-10} have essentially the same effect. As for the cover rate, a smaller α means a higher cover rate. Analogously, the gap between neighboring parameters also becomes smaller as the parameter decreases. When parameter α is selected as 10^{-9}, 10^{-10}, or 10^{-11}, they achieve the same and the best result. The best dynamic Markov model in the RockYou dataset is obtained when parameter α is 10^{-9}.

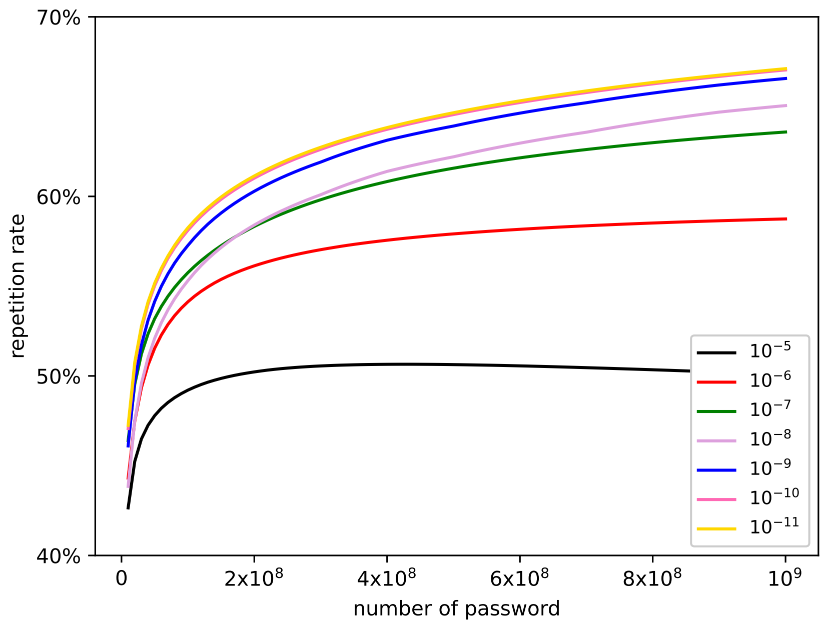

Figure 4. Repetition rate for different α values in 7 dynamic Markov models.

For the above results, we infer the core reason is that some high-probability passwords share the same n-gram fragment. After the dynamic Markov model randomly generates a high-probability password, it chooses an n-gram fragment to reduce its probability, which inevitably reduces the probability of those passwords that contain this n-gram fragment so they become low-probability passwords. This mechanism makes it difficult for these passwords to be randomly generated afterward, therefore the cover rate decreases. A larger parameter α helps generate more unique passwords. Because the probability of some important n-gram fragments is reduced too much, some low-probability passwords have more chance to be generated, but these passwords usually have the limit potential for cracking. When parameter α is selected appropriately, the password probability adjustment variation is relatively low, which can effectively prevent high-probability passwords from being submerged and reduce the repetition rate. This experimental phenomenon also shows us that the adjustment range of the password probability should be limited at a relatively flat level, so it is reasonable to randomly select one n-gram fragment instead of all n-gram fragments to adjust.

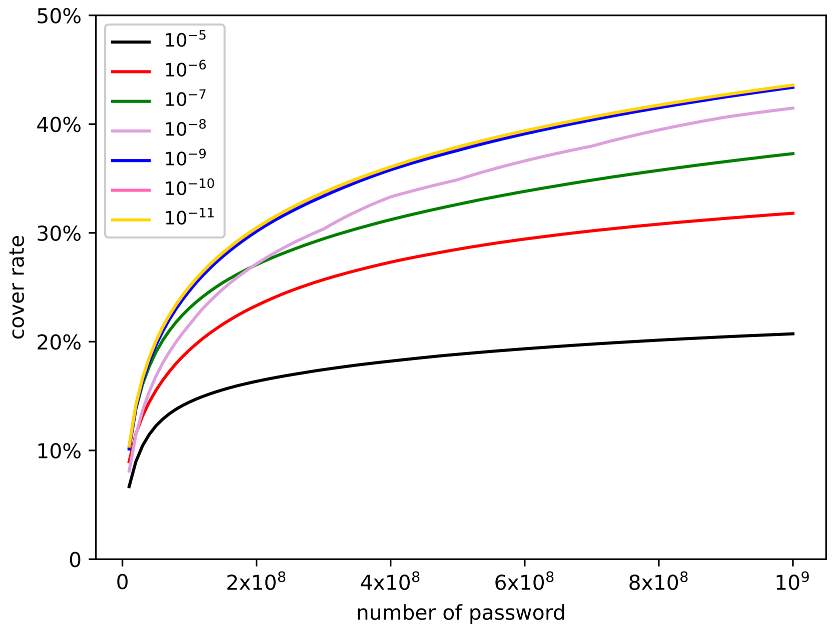

Figure 5. Cover rate for different α values in 7 dynamic Markov models.

4.2. Comparison of Performance

Then, we compare our model's performance with the standard Markov model and OMEN. In the RockYou dataset, we selected the best parameter α as 10^{-9} in the dynamic Markov model. We trained these models, generated 10^9 passwords, and calculated the CR and RR values. The experimental results are shown in Figures 6 and 7.

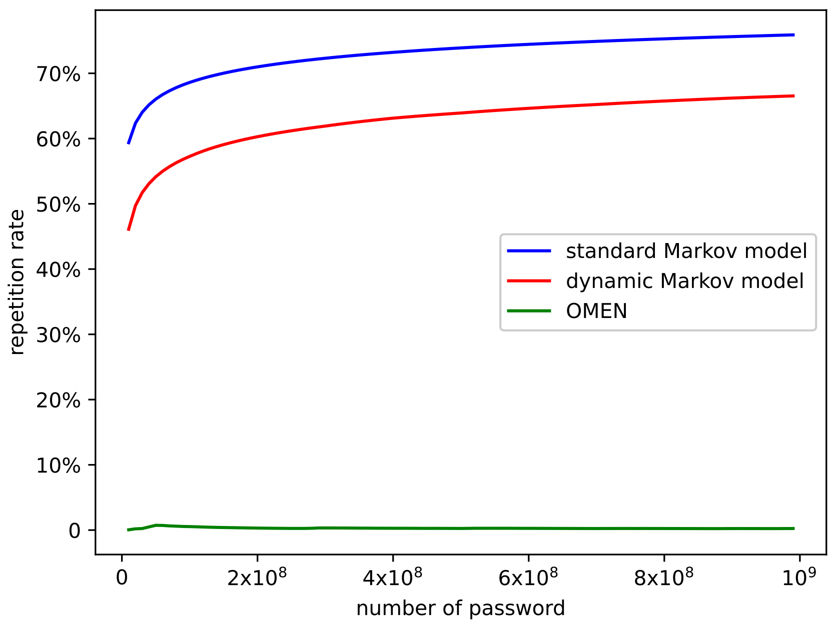

Figure 6. Repetition rate for three models.

The experimental results show that with the help of the dynamic distribution mechanism, the performance of the dynamic Markov model surpasses the standard Markov model in all aspects. The cover rate (CR) increases by 5.84% (from 37.65% to 43.49%) and the repetition rate (RR) decreases 9.38% (from 75.88% to 66.50%). Compared with OMEN, there is 13.61% gap in cover rate, and OMEN generates almost no repeated passwords.

Regarding the comparison results of the two random sampling methods, the dynamic Markov model changes the probability distribution continuously in the process of generating passwords, which finally reduces the probability value of high-probability passwords, thus reducing the repetition rate. At the same time, reducing the repetition rate would make more unique passwords appear, and hence improve the cover rate. Additionally, it is hard for an enumeration method (such as OMEN) to produce repeated passwords, therefore the corresponding *RR* is almost 0. The mechanism of generating passwords by approximate probability reduction also ensures that the cover is high enough.

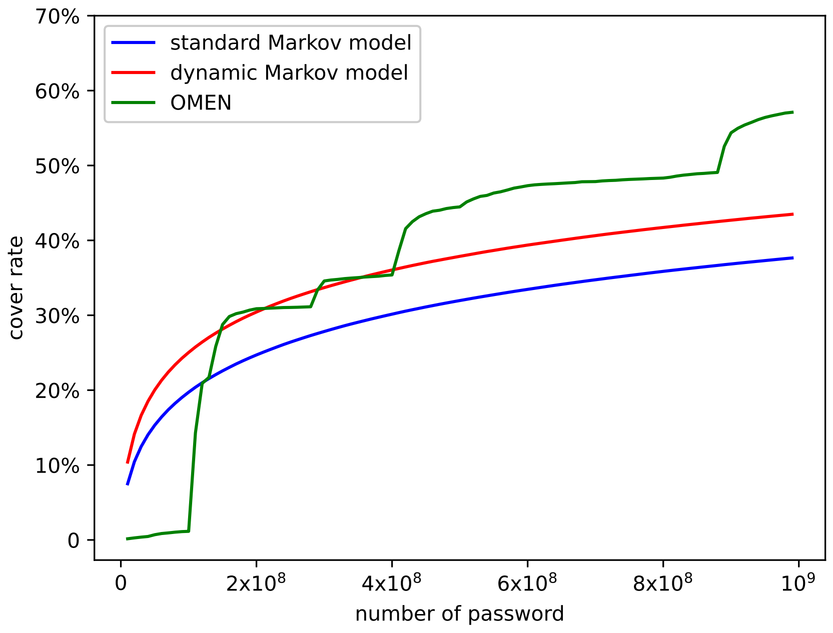

Figure 7. Cover rate for three models.

4.3. Cover Number in High Probability Passwords

Next, we compare our model's performance with the dynamic Markov model and OMEN in the high probability range. We again selected the best parameter α as 10^{-9} in the dynamic Markov model. For the test set, we calculated every password's probability by the n-gram table and divided them into 11 bins in terms of probability. We finally counted the number of hits, and the cover results are shown in Table 2.

The experimental results show that the dynamic Markov model and the standard Markov model both perform better than OMEN in the high probability range. In ranges of $10^{-4}\sim10^{-5}$, $10^{-5}\sim10^{-6}$, $10^{-6}\sim10^{-7}$, and $10^{-7}\sim10^{-8}$, random sampling achieves better cover numbers than OMEN. Especially in range $10^{-7}\sim10^{-8}$, the dynamic Markov model hits more about 16,000 passwords. In the range $10^{-8}\sim10^{-9}$, the dynamic Markov model still surpasses OMEN, but the standard Markov model fails. In regions with low probability, OMEN outperforms both the standard Markov model and the dynamic Markov model. In addition, the dynamic Markov model always works better than the standard Markov in any region.

Regarding this experimental phenomenon, we have two explanations. First, in the random sampling method, the appearance of a password is directly related to its probability value. When the number of generations is high enough, passwords with high probability will almost certainly arise, but the enumeration method generates passwords in approximately descending order of probability, which inevitably leads to neglect, so the cover number is not as good for high-probability passwords as in the sampling method. Second, based on the appropriate parameter selection, the degree of adjustment is not large, so the dynamic Markov model can not only hit high-probability passwords very well, but also tries more low-probability passwords.

4.4. Comparison of Method Determinism

Finally, we compare the three models from the perspective of algorithm determinism. After training on the RockYou training set, we ran these three generative models five times, generating 10^9 passwords each time, then compared the change of total cover rate with the number of times. The experimental result is shown in Table 3.

Table 3. Change of cover rate with generation time.

Times	Dynamic Markov	Standard Markov	OMEN
1	43.49%	37.65%	57.10%
2	49.29% (plus 5.80%)	39.11% (plus 1.46%)	57.10% (plus 0%)
3	52.59% (plus 3.30%)	39.98% (plus 0.87%)	57.10% (plus 0%)
4	54.90% (plus 2.31%)	42.18% (plus 2.20%)	57.10% (plus 0%)
5	56.88% (plus 1.98%)	43.42% (plus 1.24%)	57.10% (plus 0%)

When the dynamic Markov model was run five times, the cover rate increases by 13.39% (from 43.49 to 56.88%), which basically reaches the level of OMEN. The standard Markov model also show an improved cover rate when run more times. As for OMEN, it always generates the same password in the same order, in which there is no randomness, so the cover rate remains unchanged.

OMEN is a deterministic algorithm, so increasing the number of run times has no effect on the cover rate. However, the dynamic Markov model and standard Markov model both are stochastic algorithms, and new passwords are generated each time, so the number running of times improves the cover rate. This result also indicates that the dynamic Markov model and OMEN have the same probabilistic model regarding the passwords, but the difference lies in the order and efficiency of password generation.

5. Discussion

In the task of password guessing, the core is to build a generative model that can generate real passwords. The Markov model based on the statistical method is still widely used due to its good generation effect and high efficiency. The standard Markov model uses the random sampling method to generate passwords, which leads to a high repetition rate, thus affecting the cover rate of generated passwords. The Markov model based on enumeration is a deterministic algorithm. It always generates the same passwords in the same order and can be attacked easily in some specific cases. In addition, the cover rate of high-probability passwords by the enumeration method is not as good as the random sampling algorithm.

To address the above problems, we first design a dynamic distribution mechanism for use in the process of password generation. This mechanism continuously decreases the probability value of high-probability passwords , then the distribution of passwords continues to tend to be uniform distribution strictly, so as to achieve the effect of reducing the repetition rate and improving the cover rate. The dynamic distribution mechanism is designed based on the random sampling method, so it can avoid the shortcomings of the deterministic algorithm and enhance the cover effect of high-probability passwords. In addition, considering practical difficulties and generation efficiency, we fine-tune the dynamic distribution mechanism. We successfully apply it to the Markov model, and propose the dynamic Markov model. These changes mainly include the following: (1) randomly choose an n-gram fragment responded distribution to adjust; (2) just verify whether the probability of the n-gram fragment is greater than α; and (3) the adjustment range of probability distribution is limited to some passwords of the support set.

In the dynamic Markov model, the selection of parameter α has a significant impact on the performance. A larger parameter setting may not only significantly reduce the repetition rate, but also reduce the cover rate, and a smaller parameter setting may not be able to exert the effect of the dynamic distribution mechanism. By comparing the results of a series of experiments, we selected the optimal α value so that it would take into account both the repetition rate and the cover rate. Compared with the standard Markov model, the dynamic Markov model reduced the repetition rate from 75.88 to 66.50% and increased the cover rate from 37.65 to 43.49%. However, the cover rate still had a gap of 13.61% compared to OMEN. Through the analysis of the cover number of high-probability passwords, we find that the generation algorithm based on random sampling has better results than the method based on enumeration (OMEN), and the dynamic Markov model performed better than the standard Markov model every time. The dynamic Markov model is a random algorithm, and every password is generated randomly, while the deterministic algorithm (OMEN) does not contain any randomness and can be attacked easily. In addition, considering the effects of multiple password generation, the dynamic Markov model could achieve almost the same cover rate as OMEN.

6. Conclusions

In this work, we propose a dynamic distribution mechanism to control the repetition rate of password generation, and apply it to the Markov model based on random sampling. This model can reduce the repetition rate and improve the cover rate. Compared with the enumeration method, the algorithm keeps the advantage of randomness, and further improves the cover of high-probability passwords.

This work inspires us to improve the effectiveness of the password generative model from the perspective of reducing the repetition rate, that is, to optimize the generation efficiency as much as possible by adjusting the password probability distribution close to uniform distribution. This general idea may play a role in other password generative models. In the next research, in order to further improve the cover rate, we will consider other types of dynamic distribution mechanisms, and conduct statistical research on them, and we will also study some methods to speed up the dynamic Markov model.

Author Contributions: Conceptualization, X.G.; methodology, X.G.; software, K.T.; validation, X.G. and Y.L.; formal analysis, X.G.; investigation, M.J.; resources, M.J.; data curation, W.M.; writing—original draft preparation, X.G.; writing—review and editing, H.L.; visualization, X.G.; supervision, H.L.; project administration, H.L.; funding acquisition, H.L. and M.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by National Natural Science Foundation of China U19A2080 and U1936106, in part by the CAS Strategic Leading Science and Technology Project under Grant XDA27040303, XDA18040400 and XDB44000000, in part by the high technology project 31513070501 and 1916312ZD00902201.

Acknowledgments: This work was supported by the Beijing Academy of Artificial Intelligence, we would like to express our gratitude.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bonneau, J.; Herley, C.; Van Oorschot, P.C.; Stajano, F. Passwords and the evolution of imperfect authentication. *Commun. ACM* **2015**, *58*, 78–87. [CrossRef]
2. Herley, C.; Van Oorschot, P. A research agenda acknowledging the persistence of passwords. *IEEE Secur. Priv.* **2011**, *10*, 28–36. [CrossRef]
3. Freeman, D.; Jain, S.; Dürmuth, M.; Biggio, B.; Giacinto, G. Who Are You? A Statistical Approach to Measuring User Authenticity. In *Proceedings 2016 Network and Distributed System Security Symposium*; Internet Society: Reston, VA, USA, 2016; Volume 16, pp. 21–24.
4. Keith, M.; Shao, B.; Steinbart, P.J. The usability of passphrases for authentication: An empirical field study. *Int. J. Hum. Comput. Stud.* **2007**, *65*, 17–28. [CrossRef]
5. Florencio, D.; Herley, C. A large-scale study of web password habits. In Proceedings of the 16th International Conference on World Wide Web, Banff, AB, Canada, 8–12 May 2007; pp. 657–666.
6. Yan, J.; Blackwell, A.; Anderson, R.; Grant, A. Password memorability and security: Empirical results. *IEEE Secur. Priv.* **2004**, *2*, 25–31. [CrossRef]

7. Wang, D.; Cheng, H.; Gu, Q.; Wang, P. *Understanding Passwords of Chinese Users: Characteristics, Security and Implications*; CACR Report; ChinaCrypt: Shanghai, China, 2015.
8. Apostal, D.; Foerster, K.; Chatterjee, A.; Desell, T. Password recovery using MPI and CUDA. In Proceedings of the 2012 19th International Conference on High Performance Computing, Pune, India, 18–21 December 2012; IEEE: New York, NY, USA, 2012; pp. 1–9.
9. Dell'Amico, M.; Filippone, M. Monte Carlo strength evaluation: Fast and reliable password checking. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 158–169.
10. Zhan, X.; Hong, J. Study on GPU-based password recovery for MS Office2003 document. In Proceedings of the 2012 7th International Conference on Computer Science & Education (ICCSE), Melbourne, Australia, 14–17 July 2012; IEEE: New York, NY, USA, 2012; pp. 517–520.
11. Guddeti, P.; Dharavath, N. Analysis of password protected Document. *COMPUSOFT Int. J. Adv. Comput. Technol.* **2020**, *9*, 3762–3767.
12. Ur, B.; Kelley, P.G.; Komanduri, S.; Lee, J.; Maass, M.; Mazurek, M.L.; Passaro, T.; Shay, R.; Vidas, T.; Bauer, L.; et al. How does your password measure up? The effect of strength meters on password creation. In Proceedings of the 21st {USENIX} Security Symposium ({USENIX} Security 12), Bellevue, WA, USA, 8–10 August 2012; pp. 65–80.
13. Blocki, J.; Harsha, B.; Zhou, S. On the economics of offline password cracking. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–24 May 2018; IEEE: New York, NY, USA, 2018; pp. 853–871.
14. Zhang, M.; Zhang, Q.; Liu, W.; Hu, X.; Wei, J. TG-SPSR: A Systematic Targeted Password Attacking Model. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 2674–2697.
15. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *Adv. Neural Inf. Process. Syst.* **2014**, *3*, 2672–2680. [CrossRef]
16. Kingma, D.P.; Dhariwal, P. Glow: Generative flow with invertible 1×1 convolutions. *arXiv* **2018**, arXiv:1807.03039.
17. Salimans, T.; Karpathy, A.; Chen, X.; Kingma, D.P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv* **2017**, arXiv:1701.05517.
18. Liu, G.S.; Qiu, W.D.; Meng, K.; Li, J.H. Passward vulnerability assessment and recovery based rules minded from large-scale real data. *Chin. J. Comput.* **2016**, *39*, 454–467.
19. Narayanan, A.; Shmatikov, V. Fast dictionary attacks on passwords using time-space tradeoff. In Proceedings of the 12th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 7–11 November 2005; pp. 364–372.
20. Weir, M.; Aggarwal, S.; De Medeiros, B.; Glodek, B. Password cracking using probabilistic context-free grammars. In Proceedings of the 2009 30th IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 17–20 May 2009; IEEE: New York, NY, USA, 2009; pp. 391–405.
21. Melicher, W.; Ur, B.; Segreti, S.M.; Komanduri, S.; Bauer, L.; Christin, N.; Cranor, L.F. Fast, lean, and accurate: Modeling password guessability using neural networks. In Proceedings of the 25th {USENIX} Security Symposium ({USENIX} Security 16), Austin, TX, USA, 10–12 August 2016; pp. 175–191.
22. Hitaj, B.; Gasti, P.; Ateniese, G.; Perez-Cruz, F. Passgan: A deep learning approach for password guessing. In Proceedings of the International Conference on Applied Cryptography and Network Security, Bogota, Colombia, 5–7 June 2019; Springer: Berlin, Germany, 2019; pp. 217–237.
23. Bodkhe, U.; Chaklasiya, J.; Shah, P.; Tanwar, S.; Vora, M. Markov model for password attack prevention. In Proceedings of the First International Conference on Computing, Communications, and Cyber-Security (IC4S 2019), Chandigarh, India, 12–13 October 2020; Volume 121, pp. 831–843.
24. Vaithyasubramanian, S.; Sundararajan, R. State space classification of Markov password–an alphanumeric password authentication scheme for secure communication in cloud computing. *Int. J. Pervasive Comput. Commun.* **2021**, 17, 121–134. [CrossRef]
25. Guo, X.; Liu, Y.; Tan, K.; Jin, M.; Lu, H. PGGAN: Improve Password Cover Rate Using the Controller. In *Journal of Physics: Conference Series*; IOP Publishing: Bristol, UK, 2021; p. 012012.
26. Dürmuth, M.; Angelstorf, F.; Castelluccia, C.; Perito, D.; Chaabane, A. OMEN: Faster Password Guessing Using an Ordered Markov Enumerator. In Proceedings of the International Symposium on Engineering Secure Software & Systems, Milan, Italy, 4–6 March 2015.
27. Li, Q. A Survey Study of Password Setting and Reuse. Ph.D. Thesis, University of Delaware, Newark, DE, USA, 2020.
28. Wang, D.; He, D.; Cheng, H.; Wang, P. fuzzyPSM: A New Password Strength Meter Using Fuzzy Probabilistic Context-Free Grammars. In Proceedings of the 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Toulouse, France, 28 June–1 July 2016; pp. 595–606. [CrossRef]
29. Zipf, G.K. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*; Ravenio Books; Addison-Wesley Press: Boston, MA, USA, 2016.
30. Wang, D.; Cheng, H.; Wang, P.; Huang, X.; Jian, G. Zipf's Law in Passwords. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2776–2791. [CrossRef]
31. Wang, D.; Wang, P. The Emperor's New Password Creation Policies. In Proceedings of the Computer Security—ESORICS 2015, Vienna, Austria, 21–25 September 2015; Pernul, G., Ryan, P.Y.A., Weippl, E., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 456–477.

32. Ma, J.; Yang, W.; Luo, M.; Li, N. A Study of Probabilistic Password Models. In Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 18–21 May 2014; IEEE Computer Society: Los Alamitos, CA, USA, 2014; pp. 689–704. [CrossRef]
33. Veras, R.; Collins, C.; Thorpe, J. A Large-Scale Analysis of the Semantic Password Model and Linguistic Patterns in Passwords. *ACM Trans. Priv. Secur. (TOPS)* **2021**, *24*, 1–21. [CrossRef]
34. Teng, N.; Lu, H.; Min, J.; Ye, J.; Li, Z. PG-RNN: A password-guessing model based on recurrent neural networks. *CAAI Trans. Intell. Syst.* **2018**, *13*, 889–896.
35. Goyal, V.; Kumar, V.; Singh, M.; Abraham, A.; Sanyal, S. CompChall: Addressing Password Guessing Attacks. In Proceedings of the IEEE International Conference on Information Technology: Coding and Computing (ITCC'05), Las Vegas, NV, USA, 4–6 April 2005; pp 739–744.
36. Sprengers, M.; Batina, L. Speeding Up GPU-Based Password Cracking. In Proceedings of the SHARCS2012, Washington, DC, USA, 17–18 March 2012; pp. 35–54.
37. Bošnjak, L.; Sreš, J.; Brumen, B. Brute-force and dictionary attack on hashed real-world passwords. In Proceedings of the 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 21–25 May 2018; IEEE: New York, NY, USA, 2018; pp. 1161–1166.
38. John the Ripper Password Cracker. 2017. Available online: http://www.openwall.com/john/ (accessed on 14 May 2021).
39. Hashcat Advanced Password Recovery. 2017. Available online: https://hashcat.net/wiki/ (accessed on 13 May 2021).
40. Tansey, W. *Improved Models for Password Guessing*; Tech. Rep.; University of Texas, Austin, TX, USA, 2011.
41. Wang, D.; Zhang, Z.; Wang, P.; Yan, J.; Huang, X. Targeted Online Password Guessing: An Underestimated Threat. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security—CCS '16, Vienna, Austria, 24–28 October 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 1242–1254.
42. Zhou, H.; Wang, J.; Wang, B.; Luo, Y.; Ma, Z.; Liu, G. Comprehensive overview of plaintext password generation models. *Comput. Eng. Appl.* **2018**, 4, 9–16.
43. Houshmand, S.; Aggarwal, S.; Flood, R. Next Gen PCFG Password Cracking. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1776–1791. [CrossRef]
44. Veras, R.; Collins, C.; Thorpe, J. On the Semantic Patterns of Passwords and their Security Impact. In Proceedings of the Network & Distributed System Security Symposium, San Diego, CA, USA, 23–26 February 2014.
45. Li, Y.; Wang, H.; Sun, K. A study of personal information in human-chosen passwords and its security implications. In Proceedings of the IEEE INFOCOM 2016—IEEE Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016.
46. Sutskever, I.; Martens, J.; Hinton, G. *Generating Text with Recurrent Neural Networks*; ICML'11; Omnipress: Madison, WI, USA, 2011; pp. 1017–1024.
47. Xu, L.; Ge, C.; Qiu, W.; Huang, Z.; Lian, H. Password Guessing Based on LSTM Recurrent Neural Networks. In Proceedings of the IEEE International Conference on Computational Science & Engineering, Guangzhou, China, 22–23 July 2017.
48. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
49. Pasquini, D.; Gangwal, A.; Ateniese, G.; Bernaschi, M.; Conti, M. Improving password guessing via representation learning. *arXiv* **2019**, arXiv:1910.04232.
50. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. Improved training of wasserstein gans. *arXiv* **2017**, arXiv:1704.00028.
51. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein Generative Adversarial Networks. In *Proceedings of Machine Learning Research*; PMLR; International Convention Centre: Sydney, Australia, 2017; Volume 70, pp. 214–223.
52. Nam, S.; Jeon, S.; Moon, J. Generating Optimized Guessing Candidates toward Better Password Cracking from Multi-Dictionaries Using Relativistic GAN. *Appl. Sci.* **2020**, *10*, 7306. [CrossRef]
53. Nam, S.; Jeon, S.; Kim, H.; Moon, J. Recurrent gans password cracker for iot password security enhancement. *Sensors* **2020**, *20*, 3106. [CrossRef] [PubMed]
54. Rumelhart, D.E.; McClelland, J.L. Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*; California Univ San Diego La Jolla Inst for Cognitive Science: San Diego, CA, USA, 1987; pp. 318–362.
55. Chen, J.; Ran, X. Deep Learning With Edge Computing: A Review. *Proc. IEEE* **2019**, *107*, 1655–1674. [CrossRef]
56. Wang, P.; Wang, D.; Huang, X. Advances in Password Security. *J. Comput. Res. Dev.* **2016**, *53*, 2173–2188.
57. Srivastava, A.; Valkov, L.; Russell, C.; Gutmann, M.U.; Sutton, C. Veegan: Reducing mode collapse in gans using implicit variational learning. *arXiv* **2017**, arXiv:1705.07761.
58. Skullsecurity. RockYou. 2017. Available online: https://wiki.skullsecurity.org/Passwords (accessed on 13 May 2021).