RLPassGAN: Password Guessing Model Based on GAN with Policy Gradient

Deng Huang(✉), Yufei Wang, and Wen Chen

School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China
huangdeng@stu.scu.edu.cn

Abstract. The unsupervised neural network GAN can automatically generate synthetic samples conform to the distribution of learned samples. Therefore, password guessing models based on GAN, e.g. PassGAN are widely studied in recent years. However, there are two problems when dealing with discrete password data using GAN-based models. On the one hand, the non-differentiability of discrete password data may result in the failure of the backward of gradients; on the other hand, the outputs of the intermediate layers of the generator are incomplete password sequences, which cannot be directly evaluated by the discriminator until they reached the output layers, resulting in many redundant synthesized passwords. Therefore, a new password guessing method RLPassGAN based on SeqGAN with policy gradient are proposed in this paper. Policy gradient is applied to the proposed model to ensure that the model parameters can be continuously optimized. Furthermore, the incomplete password sequences of the output of the intermediate layers are evaluated by Monte Carlo search. The results show that in terms of the quality of the generated samples, the synthesized samples of RLPassGAN can cover more than 99% of the real passwords in the training set, while PassGAN and RNNPassGAN can only cover less than 30% of the real passwords in the training set; in terms of cracking on the specified site, RLPassGAN outperforms the two models by 16.4%–84.1%; in terms of cross-site cracking, RLPassGAN raised the cracking rate by 30.5%–84.9%.

Keywords: Password guessing · GAN · Gradient backhaul · Policy gradient · Monte Carlo search

1 Introduction

In recent years, a variety of new authentication methods have been proposed, such as biometric authentication [1], iris recognition [2], graphic password [3] and etc. However, password authentication is still the most widely utilized methods for authentication [4]. The main reason is that [5] the newly proposed authentication methods still faced some problems, such as high cost, difficult to use and deployment. Therefore, password will

Supported by: the Key Laboratory of Pattern Recognition and Intelligent Infor-mation Processing, Institutions of Higher Education of Sichuan Province (Grant: MSSB-2020-01).

W. Shi et al. (Eds.): SPNCE 2021, LNICST 423, pp. 159–174, 2022.
https://doi.org/10.1007/978-3-030-96791-8_12

remain the most important identity authentication method for a long time in the future [6].

Early password guessing relied more on "fancy" [7, 8] methods to generate password dictionaries. In 2005, Narayanan et al. [9] proposed a Markov chain-based password attacking model, which calculated the probability of password generation through the correlation relationship between characters from left to right. 2009, Weir et al. [10] proposed a password attacking model based on a probabilistic context free grammar (PCFG), in which each password is divided into letter segment L, number segment D and special character segment S. The password pattern frequency table and the character composition frequency table are counted by the model, and a set with guess frequency to simulate the probability distribution of real password is generated based on the tables. In 2016, Wang et al. [11] proposed an online targeted attacking model TarGuess based on personal information. Luo et al. [12] proposed a prediction model E-PCFG combining natural language processing (NLP) [13] and PCFG in 2017.

A series of neural network based password guessing models are proposed in recent years. In 2016, Melicher et al. [14] proposed a password guessing model based on LSTM, which predicted the next character according to the input sequence. In 2018, Xia et al. [15] proposed GenPass, which combines PCFG and LSTM to learn features from multiple password sets. The results demonstrated that it can effectively raise the cracking rate of cross-site guessing. In 2019, Hitaj B et al. [16] proposed a password generation model PassGAN,which minimizes the cross-entropy loss between generated samples and real password distribution to generate realistic guessing samples. Sungyup et al. [18] proposed RNNPassGAN in 2020. The generator and discriminator are changed from 5-layer residuals convolutional neural network in PassGAN to 1-layer RNN which is more suitable for text processing.

As it does not depend on prior knowledge, GAN-based password guessing model is considered to be a new way to replace the traditional statistical and non-parametric methods. In the GAN-based password guessing models, the parameters of generator are updated iteratively according to the backward of errors, and new candidate password samples are generated continuously through the contest between generator and discriminator. Theoretically, any differentiable function can be utilized to build generator and discriminator [17]. However, the password set contains a large number of characters and number combinations similar to "PASS +123456+@# ~", and the discrete password character data lacks continuous differentiability compared with numerical data such as pictures. Therefore, for the discrete password string data, it is difficult for the traditional GAN to effectively update the model parameters based on the discrete data through the gradient backhaul. Furthermore, it is difficult for the discriminator to evaluate the incomplete password sequence in the middle state of the generation process. These problems limit the performance improvement of GAN in the task of password guessing.

Solving the problem of GAN in gradient backhaul can effectively improve the cracking rate of relevant models in the task of password guessing. This paper makes contributions in the following aspects:

1. In this paper, a password guessing method RLPassGAN is proposed. It combines PassGAN [16] and Policy Gradient[19] in reinforcement learning. In our model,

LSTM network, which is suitable for text processing is utilized to build the generator and discriminator. In order to solve the difficult problem of gradient backhaul in traditional GAN models, both the reinforcement learning and policy gradient algorithm are utilized to update the parameters of the generated network.

2. Monte Carlo search method [20] is applied to make the password sequences in the intermediate state be complete, and thus the discriminator can generate rewards for these completed sequences. Finally, RLPassGAN is compared with two GAN-based password guessing models (PassGAN [16] and RNNPassGAN [18]).

2 Related Wrok

2.1 Password Guessing Model Based on GAN

GAN-based password guessing model, PassGAN, was applied to password security research by Hitaj B et al. [16] in 2019, which was based on IWGAN(Improved training of Wasserstein GANs) [21]. The method trains the neural network to learn the data distribution of the password, and then uses the neural network to generate a guess set with the same distribution as the training set. The generator of PassGAN imitates the distribution of the password in the training process, and the task of the discriminator is to determine whether the samples generated by the generator come from the real password sets. The process of the confrontation makes the discriminator reveal the information about the original training set, while the generator makes full use of the information to show the distribution of the original password training set, and finally optimizes the loss function through the gradient descent method. After several iterations, the discrimination ability of the discriminator is gradually improved, and the output of the generator is closer to the distribution of the original password set. 5-layer Residual Convolutional Neural Network (ResNets) is adopted in PassGAN's generator and discriminator, and its specific model structure is shown in Fig. 1.

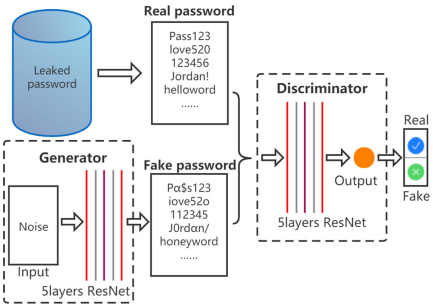

Fig. 1. The network structure of PassGAN

Sungyup et al. [18] made improvements based on PassGAN in 2020 and proposed RNNPassGAN, which utilize a 1-layer LSTM as the generator and can be more suitable for processing text. At the same time, the memory of the recurrent neural network is

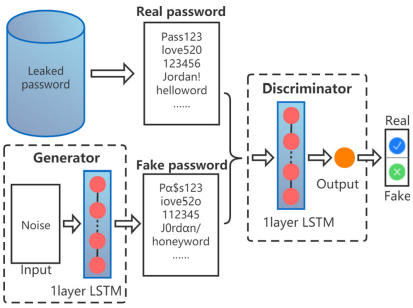

Fig. 2. The network structure of RNNPassGAN

used to improve the quality of the generated samples. The specific model structure is shown in Fig. 2.

Compared with traditional methods, GAN-based password guessing methods have the advantage that they can generate high-quality password dictionary without prior knowledge. However, they also have an obvious disadvantage that the non-differentiability of discrete password data may result in the failure of the backward of gradients, so that a good model can not be trained.

2.2 Policy Gradient

Policy gradient is an optimization algorithm in reinforcement learning, and the random strategy gradient SPG [22] is adopted in our model. In the model, the basic principle is that the agent randomly generates an action in the initial state, and a reward based on the action generated by the environment. Then the policy is dynamically adjusted by the agent based on the reward: increasing the probability of actions receiving positive rewards or decreasing the probability of actions receiving negative rewards.

$$\begin{aligned} P_\theta(\tau) =& P_\theta(s_1, a_1, s_2, a_2, \cdots, s_T) \\ =& P(s_1)\pi_\theta(a_1|s_1)P(s_2|s_1, a_1)\pi_\theta(a_2|s_2) \\ \ldots& P(s_T|s_1, a_1, s_2, a_2 \ldots s_T - 1, a_T - 1) \\ =& P(s_1)\prod_{t=1}^{T-1}(\pi_\theta(a_t|s_t)P(s_t + 1|s_1, a_1, \ldots s_t, a_t)) \\ =& P(s_1)\prod_{t=1}^{T-1}(\pi\theta(a_t|s_t)P(s_t + 1|s_t, a_t)) \end{aligned} \tag{1}$$

Policy gradient [19] contains the object system, policy $\pi_\theta\ (a|s)$, trajectory τ, round, and round-reward $R(\tau)$. The object system is the learning object of policy gradient. Policy $\pi_\theta\ (a|s)$ represents the probability of producing action a under the condition of state s and parameter θ; Round represents the process of the agent interacting with the object system from the initial state based on a policy. The track τ indicates the order of state s, action a and reward r in the policy gradient during a learning round, i.e. $\tau = (s_1,$

$a_1, r_1, s_2, a_2, r_2 \ldots, s_t$). The probability of generating track τ with parameter θ in each round is $P_\theta(\tau)$, the specific calculation formula is as formula 1. Where $P(s_1)$ represents the probability that the initial state is s_1, $P(s_{t+1}|s_t,a_t)$ represents the probability that the environment updates status to s_{t+1} based on state s_t and action a_t. According to the markov property [23], the state S_{t+1} of the next timestamp is only related to the current state s_t and the current action a_t, so the conditional probability

$$P(s_t + 1|s_1, a_1, \ldots s_t, a_t) = P(s_t + 1|s_t, a_t) \tag{2}$$

Round-reward $R(\tau)$ stands for the sum of the rewards generated by all actions in a round:

$$R(\tau) = \sum_{t=1}^{T} r(s_t, a_t) \tag{3}$$

The learning of policy gradient is a process of strategy optimization. A policy is randomly generated at the beginning, and the same policy is used in one round of learning until the end of this round of learning. Then, a new round of learning is started by changing the policy through gradient ascent, and so on until the cumulative reward no longer increases. Since the actions generated by the policy are uncertain and the same policy can produce multiple different trajectories after multiple rounds of learning, the policy gradient [19] defines the objective function as the expected return of the reward obtained by trajectory τ under strategy π_θ:

$$J(\theta) = E[R\tau|\pi_\theta] = \sum_{\tau} P_\theta(\tau)R(\tau) \tag{4}$$

The goal of training is to find a set of parameters θ to maximize $J(\theta)$, so the expected return is needed to find the partial derivative of network parameter θ, which is specifically calculated as:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \sum_{\tau} P_\theta(\tau)\nabla_\theta \log P_\theta(\tau)R(\tau) \\
&= \sum_{\tau} P_\theta(\tau)\nabla_\theta \log\left(P(s_1)\prod_{t=1}^{T-1}(\pi_\theta(a_t|s_t)P(s_t + 1|s_t, a_t))\right)R(\tau) \\
&= \sum_{\tau} P_\theta(\tau)\nabla_\theta(\log P(s_1) + \sum_{t=1}^{T-1}\log \pi_\theta(a_t|s_t) + \\
&\ldots \sum_{t=1}^{T-1}\log P(s_t + 1|s_t, a_t))R(\tau) \\
&= \sum_{\tau} P_\theta(\tau)\left(\sum_{t=1}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\right)R(\tau)
\end{aligned} \tag{5}$$

Where $P_\theta(\tau)$ indicates the probability of the generated trajectories $\tau = (s_1, a_1, r_1, s_2, a_2, r_2 \ldots, s_t)$, $\log p_\theta(\tau)$ means the logarithm of the probability of trajectory τ, and $R(\tau)$ stands for the reward of producing the entire track τ. Considering that in the expression

of $J(\theta)$, the action a_t with timestamp t has no effect on the trajectory $\tau_{1:t-1}$ before t, but just for subsequent trajectory $\tau_{t:T}$. So for $\pi_\theta(a_t|s_t)$, only the cumulative reward $R(\tau_{t:T})$ starting with the timestamp t is considered, so $J(\theta)$ transforms into:

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum_\tau P_\theta(\tau)\left(\sum_{t=1}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau_t : T)\right) \\ &= \sum_\tau P_\theta(\tau)\left(\sum_{t=1}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)Q(s_t, a_t)\right)\end{aligned} \tag{6}$$

Where $Q(s_t, a_t)$ [19] is the action-vaue function, which represents the expected return.

Value of policy π_θ after the action based on the state a_t. In the above equation, $R(\tau_{t:T})$ is substituted by $Q(s_t,a_t)$. From the above derivation, it can be seen that the policy gradient eventually transforms the value function $R(\tau)$ in the expected return into the action value function $Q(s_t,a_t)$. The advantage of this method is to identify the incomplete sequence of any intermediate state that can be evaluated by the discriminator. Finally, the parameters of network are updated as follows through gradient ascent:

$$\theta_{new} \leftarrow \theta_{old} + \eta\nabla\theta J(\theta) \tag{7}$$

Where η is the learning rate.

Policy gradient can effectively solve the difficulty of gradient return in GAN, so the password guessing model proposed in this paper, RLPassGAN, introduces the iterative training of policy gradient, and its implementation process will be introduced in the next section.

3 Reinforcement Learning PassGAN

3.1 Model Structure of RLPassGAN

Different from PassGAN [16] and RNNPassGAN [18], in our model, the proposed RLPassGAN is regarded as a reinforcement learning system, in which the generator is the agent, and each generated password sequence S_f represents a complete trajectory τ, each character comes from generating process is treated as an action, e.g. the character generated by the timestamp t is action a_t. Each action a_t is generated in the current state S_t based on a random policy, and the state S_t is determined by the generated incomplete sequence. Generator generates passwords from arbitrary characters under a set of random policy until a preset length is met. The discriminator is regarded as a reward function for the evaluation of the generated password sequences to update the neural network's parameters.

The discriminator outputs the probability of the generated password comes from the real data set as a reward, and guides the generator to adjust its policy through feedback to generate more realistic passwords. The discriminator of PassGAN only evaluates the final generated complete password. However, we want to evaluate the rewards obtained

from the incomplete sequence generated in the intermediate state during the password generation process, so as to guide the generation of subsequent characters.

In order to evaluate the incomplete sequence $Y_{1:t}$ in an intermediate state, RLPassGAN utilizes the Monte Carlo search [20] to randomly select T-t characters to padding the incomplete password sequences in the state under the guidance of the policy G_β [24]. Then, the discriminator evaluates the complete password sequence after completion, and takes the reward value as the action-value Q [19] to adjust the policy G_β. The discriminator evaluates the incomplete password sequence of all intermediate states through an iterative Monte Carlo search process until the generator produces a complete password sequence. In order to reduce deviation, the model carries out N rounds of Monte Carlo searches [20] and takes the average value of the actions returned by the discriminator. The specific search process is shown in Fig. 3, and the Monte Carlo searches are defined as:

$$\left\{Y_{1:T}^1, \ldots, Y_{1:T}^N\right\} = MC^{G\beta}(Y_{1:t}; N) \tag{8}$$

Where $Y_{1:t}$ represents the intermediate state sequence of length t, and $Y_{1:T}^N$ represents the complete sequence generated by the Nth time Monte Carlo search, which is generated by sampling under the guidance of the random policy G_β [24]. Therefore, the final action-value function of RLPassGAN is defined as:

$$Q(s = Y_{1:t-1}, a = yt) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} R\left(Y_{1:T}^n\right), Y_{1:T}^n \in MC^{G\beta}(Y1:t; N) & t < T \\ R(Y1:T) & t = T \end{cases} \tag{9}$$

Where $R\left(Y_{1:T}^n\right)$ represents the reward generated by the discriminator for the Nth time Monte Carlo search, and $R(Y_{1:T})$ represents the reward generated by the discriminator for the final generation of a complete password sequence of length T.

The generator and discriminator of the RLPassGAN are changed from 5-layer residual convolutional neural network ResNets to 1-layer LSTM, which is more suitable for processing sequence data. The network structure is shown in Fig. 3.

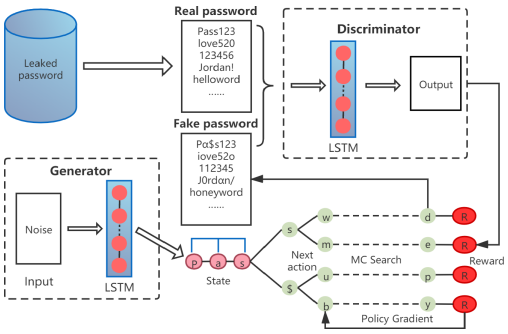

Fig. 3. The network structure of RLPassGAN

Algorithm 1 shows the steps of RLPassGAN in the training process. G_θ and G_β correspond to the random policy used in generator and Monte Carlo search. We select the

same parameters when initializing the two models, thus they can adopt the same policy to generate characters at the beginning of training. D_φ denotes the discriminator. The parameters of G_θ and D_φ are initialized with random weights before training. After the training starts, the generator randomly selects characters to generate password sequence S_f, obtains the action-value Q [19] of each intermediate transition in S_f through Monte Carlo search. Then, the reward of the complete sequence are given by the discriminator. Finally, the parameter θ is updated by generator through Formula (7) mentioned in Sect. 2. The generated guessing sample S_f and the real password S_r are input into the discriminator at the same time when the model is training the discriminator, and the reward of the guessing sample S_f is calculated and returned to the generator. The discriminator is optimized continuously by minimizing the cross entropy between the real password tag and the predicted probability. G_θ and D_φ are carried out synchronously during the whole process, and the parameters of the network are updated continuously through iterative training until the generator and discriminator are stable.

Algorithm 1 Training Process of RLPassGAN

Require: generator policy G_θ ; roll-out policy G_β; discriminator D_φ; an input password with T length: $\boldsymbol{S_r}$。

Initialize G_θ, D_φ with random weights θ, φ;
Initialize G_β with the same parameter θ
Repeat
  for g-steps **do**
    Use G_θ to generate a T length password $\boldsymbol{S_f}$
    for t in 1 : T **do**
      Compute Q ($a_t = y_t$, $s_t = \boldsymbol{S_f}$[1: t-1]) by Eq. (9)
    end for
    Update parameters of G_θ via policy gradient by Eq. (7)
  end for
  for d-steps **do**
    Use current G_θ to generate fake passwords and combine with given real passwords $\boldsymbol{S_r}$
    Train D_φ for k epochs by Eq. (10)
  end for
Reinitializes G_β based on updated θ
until RLPassGAN converges

3.2 Generator

Convolutional Neural Network takes advantage of weight sharing and the local correlation of data to reduce the complexity of the network, and it is suitable for processing continuous data such as images. Therefore, the generator of RLPassGAN is LSTM. The

generator first encodes the input random strings by word embedding to obtain the word vector of the discrete data, and then passes through a masking layer to skip the filled part of the sequence when it is encoded, so that LSTM can only act on the original part of the sequence during training. Then the model is trained by 1 layer of LSTM, and finally the password guessing set is output by a layer of full connection layer. The specific structure of the generator is shown in Fig. 4.

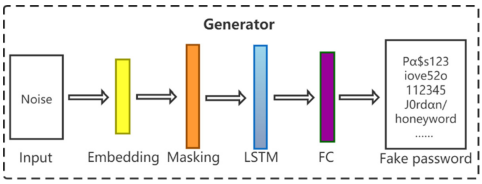

Fig. 4. The network structure of the generator of RLPassGAN

3.3 Discriminator

The discriminator of RLPassGAN is regarded as a reward function to dynamic update the parameters θ of the generator. The generator is improved by further iteration, once a more realistic set of sequences is generated, the model retrains the discriminator. Therefore, the optimization goal of the discriminator is to minimize the cross entropy between the real password tag and the predicted probability, and its specific formula is as follows:

$$\min_{\varphi}(\mathrm{E}y\sim preal[\ln D\varphi(Y)] - \mathrm{E}y\sim G\theta[\ln(1 - D\varphi(Y))]) \tag{10}$$

Where Y represents a complete password sequence, D_{φ}(Y) represents the output of the discriminator, and the parameter φ is updated at a specified learning rate to optimize the performance of the discriminator.

The structure of the discriminator is shown in Fig. 5. First, the real password set is mixed with the password set generated by the generator, and then the word vectorization is processed by an embedding layer, followed by a LSTM layer for training. In order to improve the efficiency of the discriminator's training, a highway network [25] architecture is added to the discriminator based on feature set mapping. Finally, a fully connected sigmoid activation function is used to output the probability that the generated password is the real password as the reward value. The specific structure of the discriminator is shown in Fig. 5.

4 Experiment

4.1 Password Data Set

In 2009, the SQL injection of RockYou.com by attacker resulted in 32 million user passwords being leaked, and these passwords were all plaintexts. This is the first batch

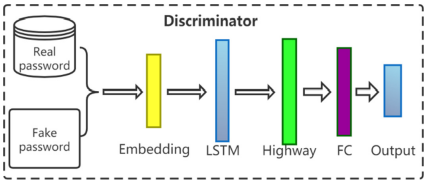

Fig. 5. The network structure of the discriminator of RLPassGAN

of ten-million-level password data sets leaked on the Internet, so these passwords have become an important resource in the field of password security research. In addition to RockYou [26], password data sets leaked from two famous Internet portals, CSDN [27] and Yahoo [28], are also used to experiments and model evaluations. Data set from CSDN [27] includes the password data of 6 million users. Since most of its users are working in the Internet industry, the password structure of the website is relatively more complex and the password security intensity is higher. Therefore, it is also more conducive for model training to improve the cracking rate and universality of the password test set. Yahoo [28] suffered a major data breach in 2012, in which 450,000 plaintext passwords of its users were stolen by hackers. The specific information of the data set is shown in Table 1.

Table 1. Password set information used in this paper.

Password set	Service type	Language	Leakage time	Original number
Rockyou	Social	English	2009	32503388
CSDN	IT BBS	Chinese	2011	6428277
Yahoo	News	English	2012	453492

4.2 Data Preprocessing

Most of the leaked password data sets are usually not stored in plaintext, but in a hash format. However, in the task of password guessing, the model needs to be trained on plaintext passwords, so the password data sets selected in this paper are all from websites that store in plaintext. Therefore, the real password sets leaked from RockYou [26], CSDN [27] and Yahoo [28] were selected as the data sets of this experiment.

Due to the large number of original data sets, it is necessary to preprocess the password sets. In the experiment, the data sets are processed by random sampling, which reduces the time cost of the experiment without affecting the experimental effect. The experiment in this paper only considers the password information in the data sets, so the irrelevant information including personal information and account number from the data sets were eliminated. At the same time, since most web sites have registration rules, users are only allowed to enter 95 printable characters (does not contain Spaces) such as

numbers, letters and symbol, and as a result of the limitation of password strength profilers, users are only allowed to enter the length of the password for 6–20, and the leaked password data sets contain many useless passwords with no reference value, Therefore, the experiment only reserves the password with the length of 6 to 20 and composed of 95 printable characters.

In this experiment, three password data sets are divided into training set and test set in a 4:1 ratio (80% is training set and 20% is test set). The training set is put into three GAN-based models for training to generate guessing set, while the test set is used to match with the guessing set and calculate its cracking rate to complete the performance comparison of the three models.

5 Evaluation

5.1 The Quality of the Generated Samples

It is proved that the model has reached an equilibrium point in GAN when the discriminator is difficult to identify the samples generated by the generator. In order to evaluate the quality of the generated samples of RLPassGAN, we compare the generation coverage rate of three models (RLPassGAN, PassGAN [16] and RNNPassGAN [18]) on different password data sets (that is, the model was trained and tested on the same data set, then calculate the ratio of the guessing samples to the repetition of the training set). The higher the generation coverage is, the stronger the fitting ability of the generator is, that is, it can generate more similar guessing samples to the training set.

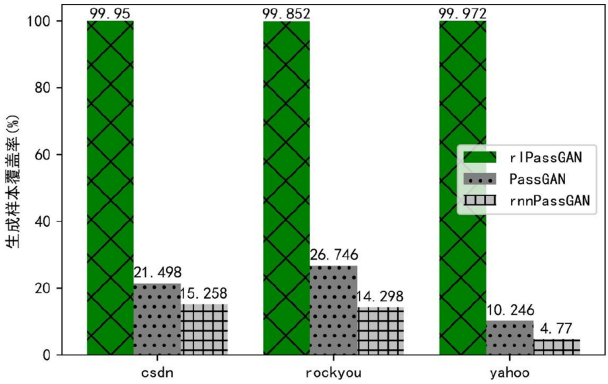

Fig. 6. Generation coverage of different models on different password sets

In order to form a comparison, we strictly controls the conditions in the experiment. No matter the training set or the guess set, the experimental data scale of different models are consistent. The quality of samples generated by different models was compared by the coverage rate of samples generated on the original training set. In order to make the experimental results more convincing, this paper conducted a comparison under different data scales(The training set scale is 10000000, 100000000 and 1000000 respectively).

Three data sets, RockYou [26], CSDN [27] and Yahoo [28] in the experiment, are randomly sampled during the training. The generator generates 10,000,000 guess samples to match the training set. Finally, take an average of the results of different size. As shown in Fig. 6. on the data set of CSDN, the guessing samples generated by RLPassGAN training can cover 99.95% of the real passwords in the training set, and the guessing samples generated by PassGAN [16] training can cover 21.498% of the real passwords in the training set. However, the guessing samples generated by RNNPassGAN [18] can only cover 15.58% of the real passwords in the training set. On the data set RockYou, the three models also achieved similar results. The guessing samples generated by the training of RLPassGAN can cover 99.852% of the real passwords in the training set, and the guessing samples generated by the training of PassGAN [16] can cover 26.746% of the real passwords in the training set. However, the guessing samples generated by RNNPassGAN [18] can only cover 14.298% of the real passwords in the training set. On Yahoo, the guessing samples generated by RLPassGAN can cover 99.972% of the real passwords in the training set, still maintaining a high generation coverage, while the guessing samples generated by PassGAN [16] can only cover 10.246% of the real passwords in the training set. However, the guessing samples generated by the training of RNNPassGAN [18] can only cover 4.77% of the real passwords in the training set.

It can be seen from the comparison experiment that no matter which data set is tested, the guessing samples generated by RLPassGAN can cover more than 99% of the real passwords in the training set, while the guessing samples generated by models PassGAN [16] and RNNPassGAN [18] do not cover more than 30% of the training set. A very important reason for this difference is that the discrete password data in the traditional GAN lead to the failure of the gradient backhaul of the discriminator when facing the password guessing task. However, the RLPassGAN proposed in this paper applies the policy gradient to solve the problem that the discrete data is not differentiable on the gradient backhaul, thus this model improves the quality of the generated samples.

5.2 Cracking Rate

In order to test the performance of a guessing method in the task of password guessing, the most simple and direct way is to bump the generated samples into the real password data set, and check the cracking rate of this guessing method. This section also compares the cracking rate of three password guessing models, RLPassGAN, PassGAN [16] and RNNPassGAN [18], on different data sets through controlled experiments, which are specifically divided into same-site cracking rate and cross-site cracking rate. The password data sets used are the three password sets introduced in Sect. 4.1.

Same-Site Cracking Rate. It is not difficult to understand that the same password data set is used in the training and testing of the same-site cracking process. In the experiment, three kinds of models are used to test on RockYou, CSDN and Yahoo. Each password data set is divided into a training set and a test set in a 4:1 ratio (The training set accounts for 80% and the test set for 20%), and the sample sizes generated by the three models after training are 10, 10^2, 10^3, 10^4, 10^5, 10^6 and 10^7. The cracking rate of each model under each generation scale is recorded.

As can be seen from Fig. 7, no matter which data set the experiment is conducted on, the cracking rate of RLPassGAN is higher than that of PassGAN [16] and RNNPassGAN [18] in the beginning. When the generation size reaches 104, the cracking rate of RLPassGAN is significantly higher than that of the other two models: The cracking rate of RLPassGAN on different password sets is 6.1%–15.32%, while that of PassGAN [16] is 0.65%–11.38%, and the cracking rate of RNNPassGAN [18] is only 0.43%–10.99%. Since the method RLPassGAN and the two models of the control experiment are proposed in the scenario of trawling attacking, rather than the target attacking, the cracking rate obtained from small-scale generated samples should not be used as the final evaluation criterion.

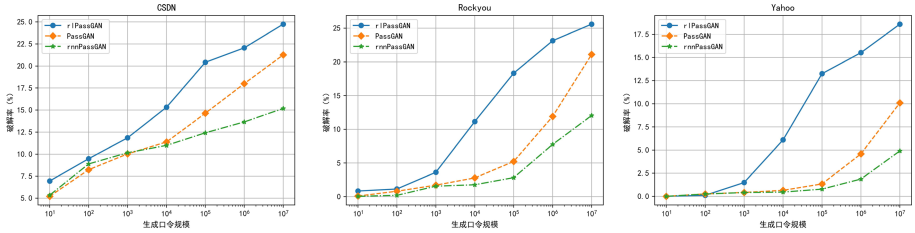

Fig. 7. The same-site cracking rate of different generation adversarial network models

With the increase of generation scale, the same-site cracking rate of the three models gradually increases, and the advantages of RLPassGAN become obvious gradually. When the generation scale is 10^5, the cracking rate of RLPassGAN on different password data sets ranges from 13.25% to 20.42%, while that of PassGAN [16] on the same-site ranges from 1.34% to 14.64%, and the cracking rate of RNNPassGAN [18] on the same-site ranges from 0.78% to 12.42%.

When the generation size reaches 10^7, each model achieves a relatively high same-site cracking rate on different password sets. The cracking rate of RLPassGAN can reach 18.6%–24.75%, and that of PassGAN [16] can reach 10.1%–21.26%, while that of RNNPassGAN [18] is only 4.91%–15.18%.

It can be seen from the experimental results that different password data sets affect the same-site cracking rate of the model, but the comprehensive performance of RLPassGAN is the best. Comparing with the cracking rate of the same-site, RLPassGAN outperform PassGAN [16] by 16.4%–84.1%. RLPassGAN can achieve a 63.1% higher cracking rate than RNNPassGAN [18]. Under the guidance of policy strategy, the RLPassGAN proposed in this paper can realize the sufficient training of GAN, which makes it better fit the distribution of real passwords, and effectively improves the cracking rate of trawling attacking.

Cross-Site Cracking Rate. In addition to compare the cracking rate of the three models on same-site attacking, we also compare that of the three models on cross-sites attacking. That is to say, the passwords of training and test come from different data sets. In practical applications, cross-site attacking is a common application scenario in the task of password guessing. Generally, it is difficult to obtain leaked plaintext passwords from

websites, so the real password data sets that can be used for research are relatively limited. Therefore, a method of password guessing with better generalization ability should not only achieve better cracking rate in the same-site guesswork, but also shows good results in cross-site guessing, which indicates that the model has good generalization ability and can be applied to different data sets. In this section, three groups of guessing experiments are conducted to compare the cross-site cracking rates of three models, RLPassGAN, PassGAN [16] and RNNPassGAN [18], as follows: RockYou → Yahoo, Yahoo → CSDN and RockYou → CSDN. The sample size of the three models is consistent with that of the same-site. The cracking rate of each model under each generation scale is recorded.

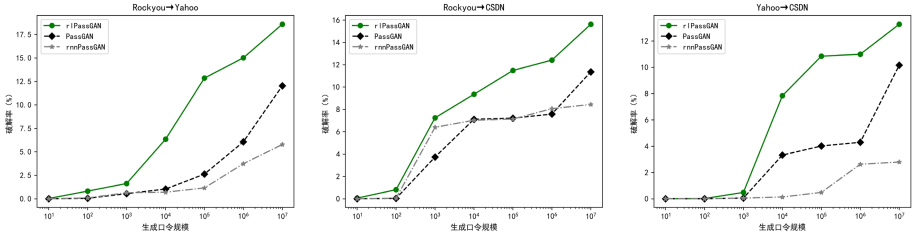

Fig. 8. The cross-site cracking rate of different generation adversarial network models

As can be seen from Fig. 8, because of the difference in the spatial distribution of the different password data sets, the cracking rate of cross-site is lower than that of same-site. However, it is found that the cross-site cracking rate of RLPassGAN is significantly better than the other two models through comparison. In the comparative experiment of cross-site guessing from RockYou to Yahoo, the cracking rate of RLPassGAN is always higher than that of the other two models, and the advantage is gradually obvious as the size of generated samples increases. When the size of generated samples exceeds 10^7, the cracking rate of RLPassGAN can reach 18.4%, while the cross-site cracking rates of PassGAN [16] and RNNPassGAN [18] are 12.04% and 5.79%, respectively. Compared with the other two models, the cross-site cracking rate of RLPassGAN is 51.2% higher than that of the better model, PassGAN [16].

In the control experiment of cross-site guessing of CSDN from Yahoo, the cracking rate of RLPassGAN reaches 13.28% after the generation scale reaches 10^7, which is 30.5% higher than that of the model with the best performance of PassGAN [16] and RNNPassGAN [18]. In the control experiment of cross-site guessing CSDN from RockYou, the cracking rate of RLPassGAN can reach 15.63%, and that of PassGAN [16] is 11.37%, while of RNNPassGAN [18] is only 8.45% with the increase of the generation scale. The cracking rate of RLPassGAN is 37.4% higher than that of PassGAN [16] and 84.9% higher than that of RNNPassGAN [18].

To sum up, RLPassGAN performs best on different password datasets in both the same-site guessing scenario and the cross-site guessing scenario. The guessing samples generated by the same model have significantly different cracking rates on different test sets. It is not difficult to understand that the three password data sets used in this paper are from different websites, and the service types and languages of websites are

not exactly the same. Therefore, the passwords constructed by each user have different characteristics, which will result in significant differences in cracking rate. The three GAN-based password guessing models involved in this paper are all studied based on trawling attacking, and the performance of RLPassGAN is better than the other two models. One of the main reasons is that RLPassGAN uses Policy gradient and Monte Carlo search to solve the difficulty of GAN when processing discrete password sequence with gradient backhaul and discriminator can not evaluate incomplete sequence.

6 Conclusion

In this paper, a new password guessing method RLPASSGAN is proposed. The policy gradient is applied to PassGAN to solve the problem of gradient backhaul caused by discrete password data. At the same time, Monte Carlo search is utilized to enable the discriminator to evaluate the incomplete password sequence at any stage. Thus, the cracking rate on the password set and the quality of generated samples are improved. In this paper, RLPassGAN is compared with two other widely utilized password guessing methods, PassGAN and RNNPassGAN. The result shows that the performance of RLPassGAN is better than the other two models in both the quality of generated samples and the cracking rate. In terms of the quality of generated samples, the guessing samples generated by RLPassGAN can cover over 99% of the real passwords in the training set. However, PassGAN and RNNPassGAN can only cover less than 30% of the real passwords in the training set. In terms of the cracking rate of the same-site, RLPassGAN is also better than the other two models. Compared with the guessing samples generated by PassGAN, the crack rate of RLPassGAN is improved by 16.4%–84.1%, and it is improved by 63.1% compared with the best performance of RNNPassGAN. In terms of cross-site cracking rate, RLPassGAN has the highest cracking rate of 30.5%–50.2% compared to PassGAN, and the best performance of RLPassGAN is 84.9% higher than that of RNNPassGAN.

For a long time in the future, password will still be one of the most important ways of identity authentication, so the study on password guessing has far-reaching significance for password security. In the future work, we will further study the characteristics of passwords, and apply the personal information and character structure in passwords to GAN, so as to improve the performance of the password guessing model.

References

1. Yang, Y., Lu, H., Liu, J.K., et al.: Credential wrapping: from anonymous password authentication to anonymous biometric authentication. In: Proceedings of ASIACCS, pp. 141–151. ACM, New York (2016)
2. Wildes, R.P.: Iris recognition: an emerging biometric technology. Proc. IEEE **85**(9), 1348–1363 (1997)
3. Biddle, R., Chiasson, S., Van Oorschot, P.C.: Graphical passwords: Learning from the first twelve years. ACM Comput. Surv. (CSUR) **44**(4), 1–41 (2012)
4. Wang, P., Wang, D., Huang, X.Y.: Advances in password security. J. Comput. Res. Dev. **53**(10), 2173–2188 (2016)

5. Bonneau, J., Herley, C., Oorschot, P., et al.: The quest to replace passwords: a framework for comparative evaluation of web authentication schemes. In: Proceedings of IEEE S&P 2012, pp. 553–567. IEEE, Piscataway (2012)
6. Herley, C., Van Oorschot, P.: A research agenda acknowledging the persistence of passwords. IEEE Secur. Priv. **10**(1), 28–36 (2012)
7. Morris, R., Thompson, K.: Password security: a case history. Commun. ACM **22**(11), 594–597 (1979)
8. Wu, T.: A real-world analysis of Kerberos password security. In: 1999 Network and Distributed System Security Symposium, San Diego, USA, pp. 13–22 (1999)
9. Narayanan, A., Shmatikov, V.: Fast dictionary attacks on passwords using time-space tradeoff. In: Proceedings of CCS 2005, pp. 364–372. ACM, New York (2005)
10. Weir, M., Aggarwal, S., De Medeiros, B., et al.: Password cracking using probabilistic context-free grammars. In: 2009 30th IEEE Symposium on Security and Privacy, 391–405. IEEE (2009)
11. Wang, D., Zhang, Z., Wang, P., et al.: Targeted online password guessing: an underestimated threat. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1242–1254 (2016)
12. Min, L., Yang, Z.: A password cracking method based on name initials shorthand structure. Comput. Eng. **43**, 188–195 (2017)
13. Veras, R., Collins, C., Thorpe, J.: On semantic patterns of passwords and their security impact. In: NDSS (2014)
14. Melicher, W., Ur, B., Segreti, S.M., et al.: Fast, lean, and accurate: modeling password guess ability using neural networks. In: The 25th USENIX Security Symposium, Austin, USA, pp. 175–191 (2016)
15. Xia, Z., Yi, P., Liu, Y., et al.: GENPass: a multi-source deep learning model for password guessing. IEEE Trans. Multimedia **22**(5), 1323–1332 (2019)
16. Hitaj, B., Gasti, P., Ateniese, G., Perez-Cruz, F.: Passgan: a deep learning approach for password guessing. In: Deng, Robert H., Gauthier-Umaña, Valérie, Ochoa, Martín, Yung, Moti (eds.) Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings, pp. 217–237. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21568-2_11
17. Goodfellow, I., Pouget-Abadie, J., Mirza, M., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
18. Nam, S., Jeon, S., Kim, H., et al.: Recurrent GANs password cracker For IoT password security enhancement. Sensors **20**(11), 3106 (2020)
19. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y., et al.: Policy gradient methods for reinforcement learning with function approximation. In: NIPS, pp. 1057–1063 (1999)
20. Browne, C.B., Powley, E., Whitehouse, D., et al.: A survey of Monte Carlo tree search methods. IEEE Trans. Comput. Intell. AI Games **4**(1), 1–43 (2012)
21. Gulrajani, I., Ahmed, F., Arjovsky, M., et al.: Improved training of Wasserstein GANs. In: Advances in Neural Information Processing Systems, vol. 30, pp. 5767–5777 (2017)
22. Ng, A.Y., Jordan, M.I.: PEGASUS: a policy search method for large MDPs and POMDPs. arXiv preprint arXiv:1301.3878 (2013)
23. Frydenberg, M.: The chain graph Markov property. Scandinavian J. Stat. 333–353 (1990)
24. Yu, L., Zhang, W., Wang, J., et al.: SeqGan: sequence generative adversarial nets with policy gradient. In: Thirty-First AAAI Conference on Artificial Intelligence, vol. 31, no. 1 (2017)
25. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv preprint arXiv:1505.00387 (2015)
26. Rockyou http://www.Rockyou.com
27. CSDN. http://www.CSDN.net
28. Yahoo. http://www.Yahoo.com