Hindawi
Security and Communication Networks
Volume 2022, Article ID 5670629, 12 pages
https://doi.org/10.1155/2022/5670629

Research Article

Password Guessing Based on GAN with Gumbel-Softmax

Tao Zhou,[1] Hao-Tian Wu,[1] Hui Lu,[2] Peiming Xu,[3] and Yiu-Ming Cheung[4]

[1]*School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China*
[2]*Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, China*
[3]*Electric Power Research Institute, China Southern Power Grid, Guangzhou, China*
[4]*Department of Computer Science, Hong Kong Baptist University, Hong Kong*

Correspondence should be addressed to Hao-Tian Wu; wuht@scut.edu.cn

Received 22 December 2021; Revised 19 January 2022; Accepted 14 March 2022; Published 27 April 2022

Academic Editor: Thi-Thu-Huong Le

Password guessing is an important issue in user security and privacy protection. Using generative adversarial network (GAN) to guess passwords is a new strategy emerging in recent years, which exploits the discriminator's evaluation of passwords to guide the update of the generator so that password guessing sets can be produced. However, the sampling process of discrete data from a categorical distribution is not differentiable so that backpropagation does not work well. In this paper, we propose a novel password guessing model named G-Pass, which consists of two main components. The first is a new network structure, which modifies the generator from the convolutional neural network (CNN) to long short-term memory- (LSTM-) based network and employs multiple convolutional layers in the discriminator to provide more informative signals for generator updating. The second is Gumbel-Softmax with temperature control for training GAN on passwords. Experimental results show the proposed G-Pass outperforms PassGAN in password quality and cracking rate. Moreover, by dynamically adjusting one parameter during the training process, a trade-off between sample diversity and quality can be achieved with our proposed model.

1. Introduction

Data security is one of the most important aspects to consider when working over the Internet. High-speed and low latency 5G networks support the emerging application technologies relying on Internet-of-Things (IoT), such as virtual reality, autopilot, and drones [1]. Meanwhile, thousands of connected devices on the IoT network will generate a tremendous amount of data, and those network traffic data involves information about the location coordinate and personal privacy. Currently, many efforts have been done toward secure data privacy protection and reliable data transmission [2], for instance, utilizing encryption algorithms to transform raw data into ciphertext, blockchain technology for secure data transmission and storage [3–5], and biometric data such as fingerprints and iris for identity verification [6, 7]. Among many security mechanisms, password-based authentication, especially text-based passwords, is a popular choice. This is partly due to the fact that password authentication depends only on people's ability to remember their passwords but does not need any additional equipment. [8, 9]. Unfortunately, users tend to choose easy-to-guess passwords associated with their birthdays or names and often use the same password on different sites and devices [10, 11]. This makes passwords vulnerable to guessing attacks and library crashing attacks. Once the attacker succeeds, a large amount of user private information will be leaked with unpredictable consequences [12–14]. For the purpose of protecting information security, it is critical to study password guessing methods.

Traditional password guessing methods can be roughly divided into two types. The first is the dictionary-based approach, which constructs a dictionary of characters and strings commonly used in passwords, and then performs multirule transformations on the dictionary entries to generate a password guessing set. Since the transformation rules need to be manually formulated, the quality of the model is strongly related to the transformation rules. The second is the probability-based approach [15, 16], in which the statistical modeling of leaked passwords is firstly

performed, so as to generate password candidates according to the probability. The probability-based approach achieves better performance than the dictionary-based approach, but it requires some prior knowledge or structural assumptions. In general, both types of approaches are unpractical for password guessing.

With the advances of deep learning, applying deep learning technology to password guessing has become a hot topic. As the most popular model in deep learning, generative adversarial network (GAN) has been successfully applied in the fields of image and video generation, image to image translation, image synthesis, and image super-resolution [17]; it works by propagating gradients back from the discriminator (denoted by D) through the generated samples to the generator (denoted by G). This is feasible when the generated data is continuous. However, a lot of data exists in the form of discrete elements, such as text and passwords, while the outputs of GAN for the discrete data generation are not differentiable, and thus, the optimization strategy based on gradient cannot be applied directly [18]. To address this problem, we propose a new password guessing model called G-Pass by using Gumbel-Softmax. The main contributions are as follows:

(1) To improve the performance of the GAN-based models, a long short-term memory- (LSTM-) based architecture is adopted in the generator of our proposed G-Pass. Moreover, we also improve the structure of the discriminator to use multiple convolutional layers with varying filter sizes, allowing the discriminator to compare real and fake passwords from different perspectives to offer the generator more diverse and comprehensive information.

(2) As far as we know, G-Pass is the first model that combines GAN and Gumbel-Softmax with temperature control for password guessing. By sampling from the Gumbel-Softmax distribution, an approximate sample from categorical distribution is generated so that the gradient-based optimizer can be used to train GAN on discrete data. In addition, we also design a strategy to dynamically adjust the temperature in the training process, allowing G-Pass to achieve a trade-off between sample diversity and quality. Otherwise, the sample diversity may be insufficient because the generator is incapable of finding all the modes of data distribution.

(3) The experiments are conducted on the RockYou data set and LinkedIn data set, and the results show that a higher password cracking rate can be achieved with G-Pass than the current GAN-based password guessing models on the 10^9 entries guessing set.

The rest of this paper is organized as follows. In Section 2, we briefly review the background and related work of password guessing. Section 3 explains the architecture of G-Pass and Gumbel-Softmax. Section 4 describes the experimental environment and related hyperparameters of G-Pass. In Section 5, we illustrate the results of controlled experiments to show the superiority of the proposed model. The final section concludes this study.

2. Background and Related Work

In this section, we review the related password guessing work, which can be summarized into the following three aspects: dictionary-based approach, probability-based approach, and deep learning approach. For convenience, the notations used in this paper are listed in Table 1.

2.1. Generative Adversarial Network. GAN [19] is one of the most widely used models in deep learning which consists of two submodels: a generator and a discriminator. They are playing a two-player minimax game during training, wherein the generator aims to generate samples that resemble those in the training set while the discriminator tries to distinguish them [20]. More formally, the two-player minimax game can be summarized as

$$f(\theta, w) = \mathbb{E}_{x\sim\mathbb{P}_r}[\log(D(x; w))] + \mathbb{E}_{z\sim\mathcal{N}(0,I)}[\log(1 - D(G(z; \theta); w))]. \tag{1}$$

Since Goodfellow [19] proposed the first GAN model, many variants with better performance have been created. Wasserstein GAN (WGAN) [21] and WGAN with gradient penalty (WGAN-GP) [22] are two GAN models that provide stability, enabling the GAN model to locate the global optimum. WGAN-GP, introduced by Gulrajani, adopts a gradient penalty instead of gradient clipping to achieve steady training of the GAN model. It has been demonstrated that text generation tasks can be accomplished by the WGAN-GP model.

2.2. Related Work. Traditional password guessing methods can be broadly divided into two types. The first is the dictionary-based approach, such as Hashcat and John the Ripper. In general, passwords are transmitted and stored in the form of hash codes rather than plaintext, and these two tools can increase the computation speed of hash codes with the help of GPU [23]. To crack a password, it must start with a dictionary of passwords and then compare the hash codes of each entry in the dictionary to locate the target password. In other words, if the target password is not in the dictionary, it cannot be cracked. Therefore, these two tools adopt multirule transformations to expand the dictionary, such as reversing, appending, and truncating. The second is the probability-based approach, of which Markov [24] and Probabilistic Context-Free Grammar (PCFG) [16] are two representative approaches. Markov models the relationship between the adjacent characters in passwords by calculating the probability of the next character based on the preceding context. PCEG models the structure of passwords, in which the password is divided into three different tagged-label, i.e., alphabetic segment L, numeric segment D, and special character segment S. For instance, password "Password123!" is transformed to L8D3S1 in PCFG. This tagged-label

Table 1: List of notations used in the study.

Notation	Description
$\mathbb{E}$	Expected value
$\mathbb{P}_r$	Distribution of real data
D	Discriminator
$x \sim \mathbb{P}_r$	Variable x follows the distribution $\mathbb{P}_r$
w	Parameters of discriminator
$\mathcal{N}(0, I)$	Standard normal distribution
G	Generator
θ	Parameters of generator
$\mathbb{R}^v$	v-dimensional real numbers vector set
$h \in \mathbb{R}^v$	h is An element of the set $\mathbb{R}^v$
softmax	$\text{softmax}(h) = (\exp(h_i)/\sum_{j=1}^{v} \exp(h_j))$, where h is an v-dimensional vector
one_hot (i)	one-hot is a group of bits among which the legal combinations of values are only those with a single high (1) bit in index i and all the others low (0)
argmax	The points, or elements, of the domain of some function at which the function values are maximized
$/_2$	L^2 norm
G_θ	Generator with parameters θ
D_w	Discriminator with parameters w
∇_θ	The vector of partial derivatives with respect to parameters θ
∇_w	The vector of partial derivatives with respect to parameters w
$U[0, 1]$	Standard uniform distribution

structure is used to analyze and calculate the distribution probability. Finally, PCFG generates a password by choosing the tagged label with the highest probability.

With the significant growth in computing power, deep learning has become the third method of password guessing. Compared to the traditional password guessing methods, it has a larger password space and the generated samples are not limited to the training set. Based on the structure of models, deep learning-based methods can be separated into two types, RNN-based password guessing [25, 26] and GAN-based password guessing [27–29]. Melicher et al. [25] first applied deep learning to password guessing, introducing a password guessing model based on recurrent neural network (RNN). Xu et al. [26] proposed an LSTM-based password guessing model. Compared with standard RNN, LSTM can capture long-term dependencies in passwords and thus achieve better performance. Xia et al. proposed hybrid models that integrated RNN and PCFG, such as PL (PCFG + LSTM) and PR (PCFG + RNN) [23]. Compared with PCFG, Markov, and RNN models, the cracking rate of hybrid models is higher. Hitaj et al. first applied GAN in password guessing and proposed the PassGAN [27] in 2019. Sungyup proposed rPassD2SGAN [28] based on PassGAN with improvements in two aspects. The first is to replace the CNN of PassGAN with RNN, and the other is to employ the structure of double discriminators (two discriminators D1 and D2) [30]. Moreover, rPassD2SGAN adopts a new training strategy that trains D1 and D2 step by step instead of training both D1 and D2 in one step to make the model training more stable. The experiments showed that rPassD2SGAN achieves 10%–15% improvement over PassGAN.

2.3. Choice of Neural Network. A convolutional neural network (CNN) is suitable for processing image data but not the ideal choice for passwords containing sequential information. RNN is well suitable in the field of password guessing so that LSTM [31] is employed in the proposed G-Pass Generator as an RNN cell type. Considering that CNN achieves better performance in text classification tasks [32], we use CNN structure in the discriminator of G-Pass.

3. Proposed Model

In this section, we explore how to improve password cracking performance from two perspectives. The first is to redesign the architecture of the model to obtain better sample quality, and the second is to adopt a new sampling strategy to achieve the balance between sample diversity and quality.

3.1. Overall Framework. WGAN-GP [22] is employed to instantiate PassGAN, in which CNN is used to construct the discriminator and generator, as shown in Figure 1. CNN is an expert in processing image data but not the best option for passwords [28]. Traditional RNN suffers the problem of long-term dependencies due to gradient vanishing or gradient explosion, while LSTM [31, 33] is a new type of RNN that adds gate units to address this problem. It has been applied in some text generation studies [34, 35] and proved to be well suitable for handling sequential information. To improve password quality, the generator structure is switched from CNN to LSTM.

The discriminator used in PassGAN is a CNN-based classifier that employs convolutional layers with filters of the same size to capture relations of inputs. However, in the proposed model G-Pass, we adopt a new discriminator structure that uses multiple convolutional layers with filters of different sizes to capture relations of various lengths and a max-pooling layer over the outputs of those convolutional

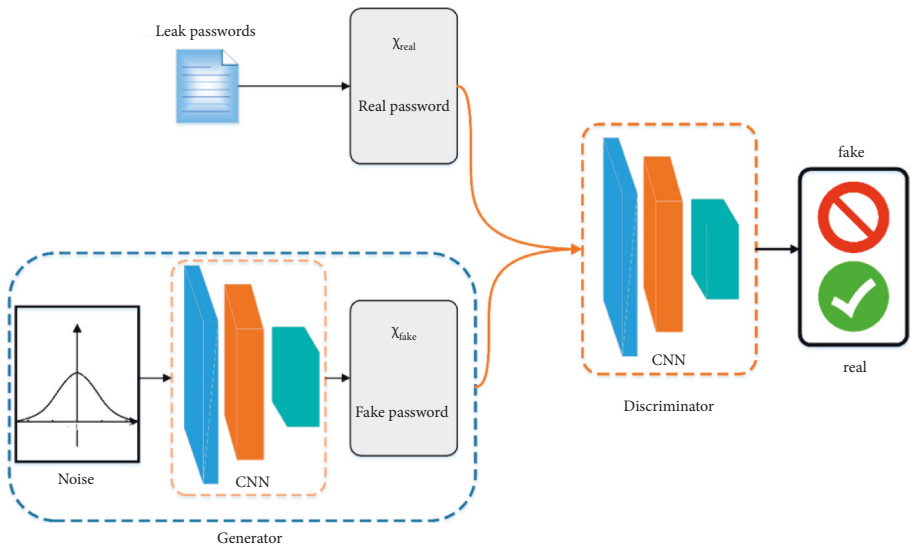

Figure 1: The architecture of PassGAN.

layers. With each input being fed into the above convolutional layers separately to get an individual feature map, the weight of those feature maps serves as the final guiding signal to update the generator. Our rationale is that filters of varying sizes can capture unique information about the input passwords, allowing the discriminator that compares real and fake passwords from these perspectives to offer the generator more diverse and comprehensive information. In this discriminator architecture, the input is a sequence of length l represented by a matrix $X \in \mathbb{R}^{l \times v}$, where each row $x_i \in \mathbb{R}^v$ is a v-dimensional vector that represents the probabilities of words in the vocabulary. The overall structure of G-Pass is shown in Figure 2, where details of the discriminator are presented in Figure 3.

3.2. Gumbel-Softmax. In the process of GAN training, the generator is used to create a fake sample and then feed it to the discriminator to calculate the loss for updating. Since passwords are discrete data, we need to obtain samples from a categorical distribution. The softmax function can be used to parameterize a categorical distribution, assuming that the vocabulary size is $\mathbf{v}$ and $\mathbf{h} \in \mathbb{R}^v$ is the output of the last layer of the generator; P is a v-dimensional vector of probabilities specifying the categorical distribution by

$$P = \text{softmax}(h). \tag{2}$$

Let $\mathbf{y}$ be a categorical variable that follows the categorical distribution $P\ (p_1, p_2, \ldots, p_v)$, where p_i is the word probability calculated by softmax. Assume our passwords are encoded as one-hot vectors. The most common way of sampling $\mathbf{y}$ is given by

$$y = \text{one\_hot}\left(\max_i (p_1, p_2, \ldots, p_v)\right). \tag{3}$$

But this sampling process is not differentiable because of the max function. In order to obtain a differentiable approximation, a simple method is to pass the probabilities calculated with softmax to the discriminator directly, without performing any sampling operations, which is called S-Pass in this work. Another method is Gumbel-Max. It has been proved [36] that sampling $\mathbf{y}$ from the categorical distribution by Equation (3) is identical to sampling $\mathbf{y}$ according to

$$y = \text{one\_hot}\left(\arg\max_i (h_i + g_i)\right). \tag{4}$$

This is a reparameterization trick, which refactors the stochastic sampling process as a combination of a deterministic ($\mathbf{h_i}$) and a stochastic ($\mathbf{g_i}$) element. Here, $\mathbf{g_i}$ is a variable that follows a standard Gumbel distribution with the cumulative distribution function as

$$F(x) = e^{-e^{-x}}. \tag{5}$$

And the inverse function is

$$x = F^{-1}(u) = -\log(-\log(u)). \tag{6}$$

According to the inverse transform sampling, we can calculate that

$$g_i = -\log(-\log(u)), \tag{7}$$

where $\mathbf{u} \sim U[0, 1]$. Since the one_hot(argmax$(\cdot)$) is still nondifferentiable, we use the softmax function as a differentiable approximation. Now, the gradient can be propagated to the discriminator along the deterministic element $\mathbf{h}$, while the sample vector $\mathbf{y}$ is given by

$$y = \text{softmax}(\tau(h + g)). \tag{8}$$

The distribution defined by Equation (8) is called the Gumbel-Softmax distribution, which was independently discovered by Maddison et al. [37]. A GAN on discrete data can be trained by using Equation (8).

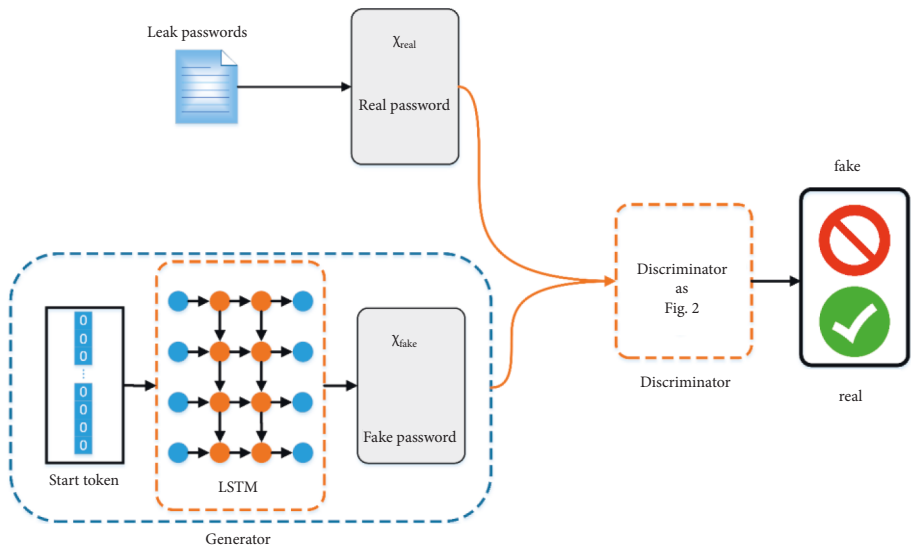

Figure 2: The architecture of the proposed G-Pass.

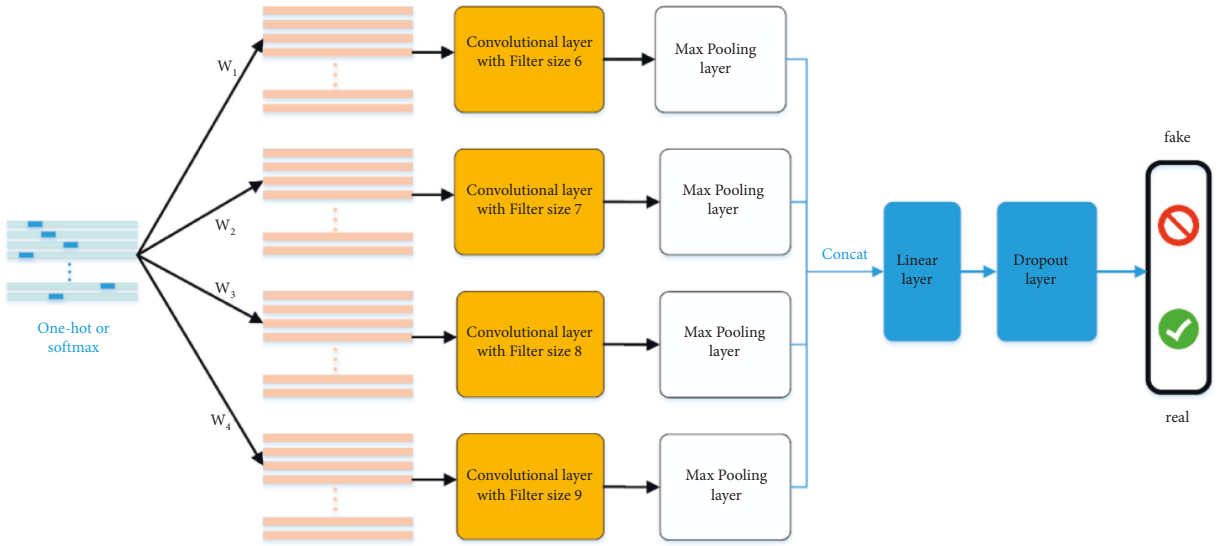

Figure 3: The discriminator in the proposed G-Pass uses multiple convolutional layers with variable size filters to capture specific information about the input. Since the length of the input is padded to 10, we set the filter sizes of the convolutional layers to 6, 7, 8, and 9, respectively.

3.3. Temperature Control Strategy. According to the formula

$$\mathrm{Var}(bx) = b^2\mathrm{Var}(x), \tag{9}$$

the $\mathrm{Var}(y) \propto \tau^2$, where $\mathrm{Var}(\cdot)$ represents the variance of a distribution. Meanwhile, the variance of the gradient $\mathrm{Var}(\partial y/\partial h) \propto \tau^2$. The higher the $\tau$is, the larger the variance of the distribution is, which is helpful to improve the diversity of samples. In contrast, with a larger variance of the gradient $(\mathrm{Var}(\partial y/\partial h) \propto \tau^2)$, the parameter updates will become very sensitive to the input. Intuitively, this might cause poor sample quality. The lower the τ is, the stabler the gradient is, which is good for robust training of the model. However, due to the small variance of the distribution $(\mathrm{Var}(y) \propto \tau^2)$, the diversity of samples may be insufficient. For positive and suitable values of $\boldsymbol{\tau}$, the model can achieve a balance between sample quality and sample diversity. Therefore, a larger $\boldsymbol{\tau}$ motivates more breadth to increase the diversity of samples, whereas a lower $\boldsymbol{\tau}$ encourages more depth for better sample quality. For these reasons, we design a strategy to dynamically adjust $\boldsymbol{\tau}$ in the training process: $\tau = \tau_{\max}^{n/N}$, where $\tau_{\mathbf{max}}$ represents the maximum temperature, N is the maximum of training iterations, and n is the current iteration. Initially, a relatively small $\boldsymbol{\tau}$ is chosen and then increased to $\tau_{\mathbf{max}}$ during training [38]. Note that continuous vector $\mathbf{y}$ is only used during training. The sample vectors are discretized to one-hot vectors by "arg max" during evaluation.

The whole training process of G-Pass is presented in Algorithm 1, in which hyperparameters λ, N, n_{critic}, m, l_r,

β_1, β_2, and τ_{max} need to be set. The parameters' setting in our experiments will be specified in Section 4. In addition, the algorithm optimizer and the loss function are Adam optimizer and Wasserstein loss function with gradient penalty, respectively. Adam optimizer is computationally efficient with little memory requirements, well suited for problems that are large in terms of data or model parameters [39]. Meanwhile, the Wasserstein loss function with gradient penalty has been used in many GAN models. With the gradient penalty term restricting the gradient in a certain range, stable training and fast convergence of GAN models can be obtained.

4. Experiment Setup

In this section, we first discuss the datasets chosen in experiments and then give the model hyperparameters settings and detailed platform configurations.

4.1. Data Set. Password guessing experiments were implemented on the passwords leaked from RockYou and LinkedIn as both of them have been widely used in related studies. As more than 60% of entries have lengths of 6 to 10, we focus on guessing passwords with lengths between 6 and 10 by randomly choosing 6 million passwords from RockYou with lengths in the range. For the passwords with lengths of less than 10, <PAD> was appended at their ends. Among RockYou passwords, 80% were used as the training set; for the remaining 20%, nonrepeating passwords that were not included in the training set were used as a test set. In addition, all nonrepeating LinkedIn passwords with lengths of 6 to 10 that were not included in the RockYou training set were used as another test set. Finally, we got a training set consisting of 4800000 entries and two test sets from RockYou and LinkedIn consisting of 1196874 and 2560463 entries, respectively. For other untested datasets, the main reason for not using them is that there are few studies implemented on these datasets, making it difficult to obtain the performance of other password guessing models on these datasets as well.

4.2. Hyperparameters. We experimented with a range of hyperparameters to take advantage of GAN to estimate the distribution of passwords from the training set. Here, are some hyperparameters used in the model.

(1) **Number of iterations**, which indicates the time of the model calls its forward propagation and backpropagation. We set the number of iterations to 99,000 in G-Pass.

(2) $\mathbf{n_{critic}}$, this value represents how many iterations the discriminator performs in each generator update. The number is set to 5 in G-Pass.

(3) **Batch size** $\boldsymbol{m}$, which represents the number of passwords obtained from the training set that propagate through the GAN at each update of the model. It is set to 64 in G-Pass.

(4) **Gradient penalty coefficient** λ, which specifies the gradient penalty weight applied to the norm of the gradient. We set the value of the gradient penalty to 10 in G-Pass.

(5) **Adam optimizer's hyperparameter: Coefficients $\boldsymbol{\beta 1}$ and $\boldsymbol{\beta 2}$** of the Adam optimizer are set to 0.5 and 0.9, respectively. The **learning rate l_r** is set to 10^{-4}. These parameters are the default values used by PassGAN.

(6) **Maximum of temperature** $\tau_{\mathbf{max}}$, this parameter is not required in the PassGAN and only exists in our proposed G-Pass, which was set to 1, 2, 5, and 10 in the controlled experiments, respectively.

The PassGAN and our proposed S-Pass and G-Pass were implemented with the PyTorch version 1.7.1 and the *Python* version 3.85. All of the experiments were run on the Ubuntu 16.04.7LST with 64 GB RAM, an Intel(R) Xeon(R) E5-2620 v4@2.10GHz 32 Core CPU, and two NVIDIA GeForce GTX 1080Ti GPUs with 11 GB memory.

5. Performance Evaluation

In this section, we evaluate the performance in three metrics. The first metric is the cracking rate, which is defined to reflect the accuracy of model guessing.

$$\text{Cracking rate} = \frac{\text{number of match passwords}}{\text{size of test set}} \times 100\%, \quad (10)$$

where the match passwords are the passwords that appear in the test set. The second is used to detect the diversity of samples, measured by the number of unique passwords created by the generator. The third is proposed to evaluate the quality of the samples, which is defined by

$$N_{\text{match}} = \frac{\text{number of unique passwords}}{\text{number of match passwords}}, \quad (11)$$

where the unique passwords are the nonrepeating passwords generated by the model, and the average number of unique passwords required to generate a match password can be calculated by N_{match} . The larger the N_{match} is, the worse the sample quality is.

5.1. Impact of Gumbel-Softmax. The RockYou training set is used to train PassGAN [27], S-Pass, and G-Pass. The hyperparameters of S-Pass are exactly the same as those in G-Pass to ensure experimental validity. The obtained results are shown in Table 2, wherein the cracking rate of G-Pass is 2.25% higher than that of PassGAN. In contrast, the performance of S-Pass is 6% lower than PassGAN. Gumbel-Softmax is employed in G-Pass to approximate "arg*max* sampling" in the training process. In contrast, S-Pass does not perform any sampling operation in the training process. This leads to poor sample diversity when using S-Pass, resulting in a lower cracking rate.

To deeper explore the cracking performances of these models, we also calculated the cracking rate on guessing sets with varying sizes. As shown in Figure 4, the cracking rate of

(i) **Input:** The gradient penalty coefficient λ, the maximum of training iterations N, the number of discriminator iterations per generator iteration n_{critic}, the batch size m, the learning rate $l-r$, Adam hyperparameters β_1 and β_2, initial discriminator D parameters w_0, initial generator G parameters θ_0, the highest temperature $\tau_{\max}$
(1) **for** $n \longleftarrow 1$ to N do
/* training discriminator */
(2) **for** $t \longleftarrow 1$ to $n_{crictic}$ **do**
(3) **for** $i \longleftarrow 1$ to m **do**
(4) Sample real data $x \sim \mathbb{P}_r$, $Z = start\_token$, random number $\alpha \sim U[0,1]$, τ is the temperature of current iteration;
(5) $y \longleftarrow G_\theta(Z, \tau)$;
(6) $x \longleftarrow \alpha x + (1-\alpha)y$;
(7) $L^{(i)} \longleftarrow D_w(y) - D_w(x) + \lambda(\|\Delta_x D_w(x)\|_2 - 1)^2$;
(8) end
(9) $w \longleftarrow \text{Adam}(\nabla w 1/m \sum_{i=1}^{m} L^{(i)}, w, l\_r, \beta_1, \beta_2)$.;
(10) end
/*training generator */
(11) Sample a batch of input of generator $\{z^{(i)}\}_{i=1}^{m} = start\_token$;
(12) $\theta \longleftarrow \text{Adam}(\nabla \theta 1/\text{m} \sum_{i=1}^{m} -D_w(G_\theta(z^{(i)}, \tau), \theta, l\_r, \beta_1, \beta_2))$
/* update the temperature */
(13) $\tau = \tau_{\max}{}^{n/N}$
(14) end

Algorithm 1: G-Pass with gradient penalty.

Table 2: Cracking rates of PassGAN [27], S-Pass, and G-Pass with guessing set size of 10^9.

Model	Guessing set	Cracking rate (%)
PassGAN [27]	10^9	14.81
S-Pass (without Gumbel-Softmax)	10^9	8.81
G-Pass ($\tau_{\max}$ =1)	10^9	17.06

each model increases slower with larger guessing sets. Meanwhile, the performance gap between G-Pass and PassGAN is also enlarged. It can be seen that the cracking rate of G-Pass will far exceed that of PassGAN as the guessing set is expanded.

Table 3 lists the number of unique passwords, match passwords, and the corresponding N_{match} of guessing set with a size of 10^9. In terms of sample diversity, the unique passwords created by G-Pass are more than those by S-Pass, demonstrating that Gumbel-Softmax adopted in the training process is beneficial to improve the diversity of samples. However, G-Pass is much less diverse than PassGAN, which is due to the fact that the generator inputs of PassGAN are random noises. In contrast, the generator inputs of G-Pass are fixed start tokens, resulting in fewer sample patterns of G-Pass than PassGAN. In terms of sample quality, PassGAN has a larger N_{match} compared to S-Pass and G-Pass, indicating that more unique passwords are required to generate a match password and the quality of samples generated by PassGAN is lower than those of S-Pass and G-Pass.

5.2. Impact of Temperature. We also test the trade-off between sample quality and diversity controlled by maximum temperature $\tau_{\max}$. Four controlled experiments were implemented with different $\tau_{\max}$, 1, 2, 5, 10. The cracking rates of the G-Pass with those temperatures are shown in Figure 5. It can be seen that the performance of the model gradually increases as $\tau_{\max}$ grows under the condition of $\tau_{\max} < 10$. But the performance at $\tau_{\max} = 10$ was worse than the case of $\tau_{\max} = 5$. This is due to the fact that the increments obtained from the growth of sample diversity are less than the reductions caused by the decrease of sample quality, leading to the bad performance of G-Pass at $\tau_{\max} = 10$.

To prove this hypothesis, the number of unique passwords and N_{match} on guessing sets of varying sizes are illustrated in Figures 6(a) and 6(b), respectively. As $\tau_{\max}$ grows, the unique password increases, which implies better sample diversity. But N_{match} also increases when the guessing set is larger enough, which indicates the worse sample quality. Table 4 shows the results of the guessing set with a size of 10^9, in which the number of unique passwords of G-Pass ($\tau_{\max} = 10$) grows by 8.65% compared to that of G-Pass ($\tau_{\max} = 5$). Meanwhile, N_{match} increases by 16%, leading to a 1.67% drop in cracking rate.

5.3. Performance Evaluation under Different Password Lengths. To compare the performance under different passwords lengths, we counted the length of passwords in the RockYou test set and the cracking rate of PassGAN and G-Pass under different password lengths. The results are shown in Tables 5 and 6, respectively. It can be seen that the cracking rate of G-Pass is higher than that of PassGAN for

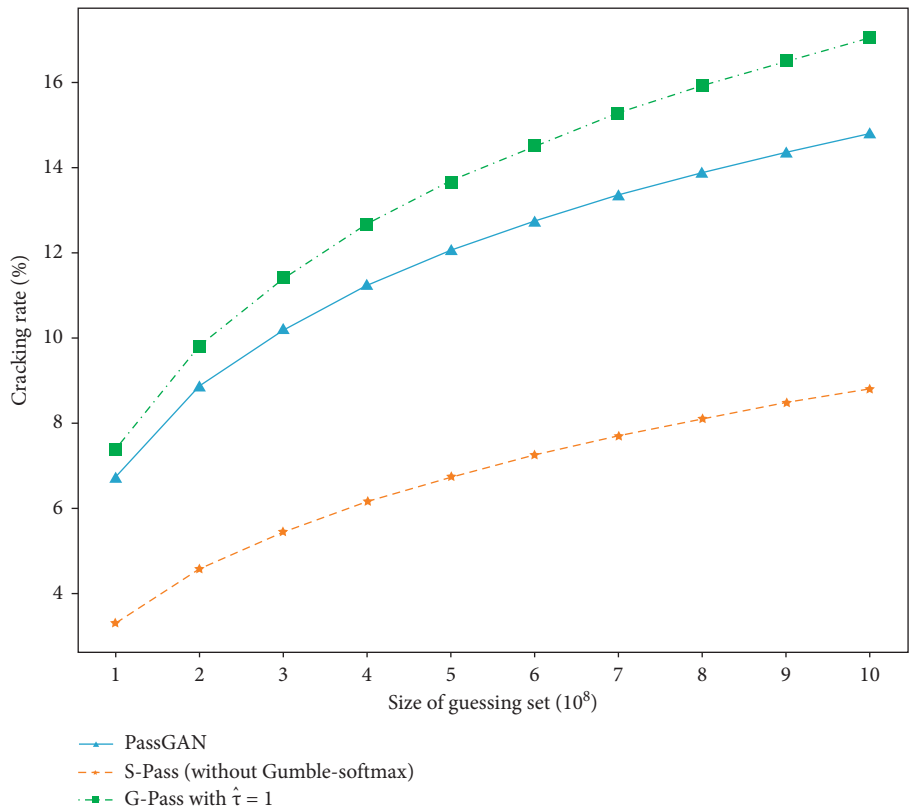

Figure 4: Cracking rates of PassGAN [27], S-Pass, and G-Pass with respect to the size of guessing sets.

Table 3: Performance of PassGAN [27], S-Pass, and G-Pass with a guessing set size of 10^9.

Metric	PassGAN [27]	S-Pass (without Gumbel-Softmax)	G-Pass ($\tau_{\max} = 1$)
Unique passwords	323,337,485	115,221,059	170,609,130
Match passwords	177,257	105,445	204,172
N_{match}	1,824	1,092	836

different lengths. The cracking rate increases from 29.96% to 45.06% ($\tau_{\mathbf{max}} = 5$) when password length is 6. In contrast, the cracking rate grows only 6.19% in cases with a password length of 10. There are two main reasons for the high cracking rate of shorter passwords. The first is that the shorter passwords in the test set are almost combinations of lowercase letters and numbers which are vulnerable to guessing attacks. In contrast, long passwords contain many complex password combinations such as mixed-letter passwords (e.g., lOvejaacup) and lee speak passwords (e.g., mem@rry!, ladyluck<3, maggz4life, kristina<3). Therefore, to force users to strengthen their passwords, many sites utilize a program to check the validity of a password, which must contain at least one lowercase letter, one uppercase letter, and one special character. The second reason is that it is easier to crack short passwords than long ones. For a cracker, it is exponentially less likely to crack a password as its length increases. Moreover, the relative improvement with the proposed G-Pass over PassGAN [27] is listed for different lengths of password and $\tau_{\max}$, respectively. Among them, the highest improvement is achieved with G-Pass ($\tau_{\mathbf{max}} = 5$), while generally10-15% better performance can be obtained with rPassD2SGAN [30]. It can be seen that much better performance can be achieved by choosing the appropriate $\tau_{\mathbf{max}}$ in applying the proposed G-Pass.

5.4. Test on Different Data Sets. To verify the generalization ability of PassGAN and the proposed G-Pass, we also tested them on the LinkedIn test set. The results are shown in Figure 7, where an increase in the cracking rate can be achieved with the proposed G-Pass (e.g., 0.36% for $\tau_{\max} = 1$ and 5.81% for $\tau_{\max} = 5$). From the cracking rates listed in Table 7, it can be seen that G-Pass ($\tau_{\max} = 5$) also performs the best on the LinkedIn test set. Note that the cracking rate of each model on the LinkedIn test set is lower than that on the RockYou test set. This is because RockYou and LinkedIn

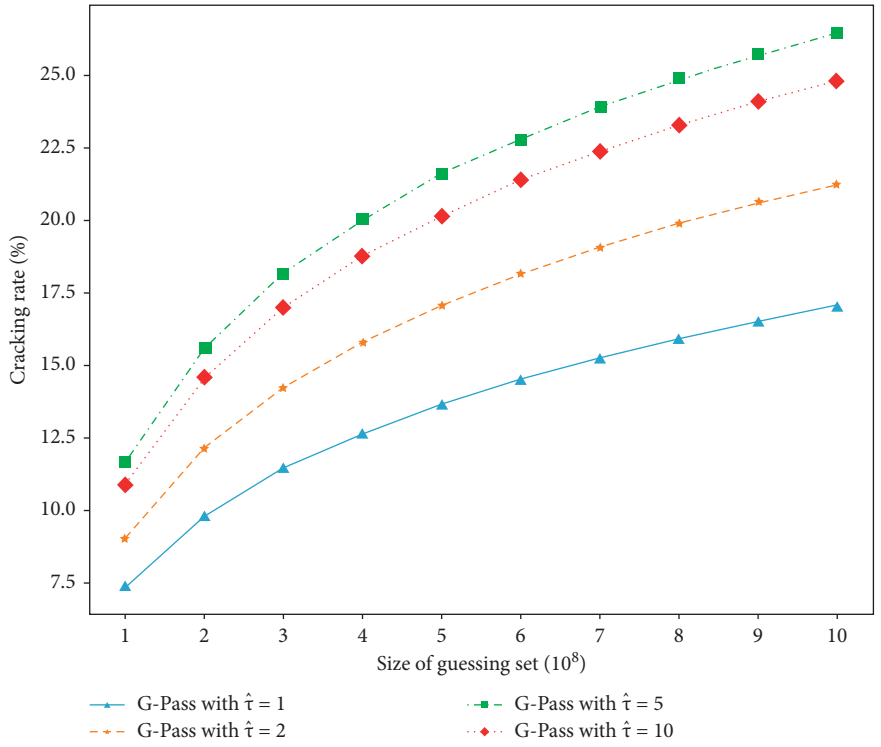

Figure 5: Cracking rate of the proposed G-Pass with respect to $\tau_{\max}$.

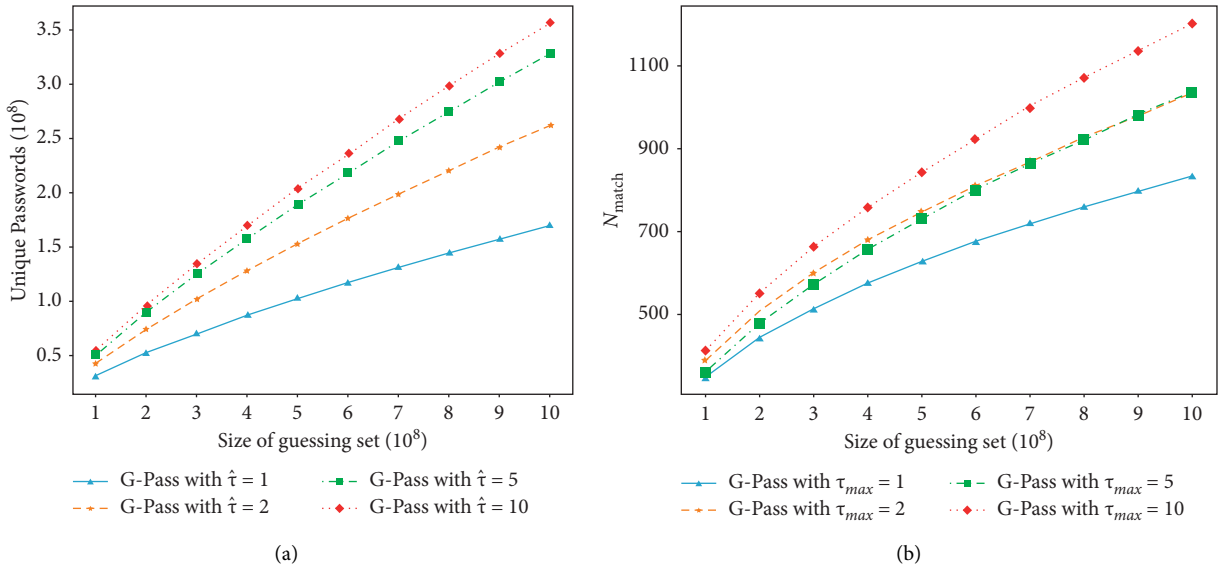

Figure 6: Performance of the proposed G-Pass with respect to $\tau_{\max}$. (a) Unique passwords. (b)N_{match}.

Table 4: Performance of G-Pass with different values of $\tau_{\max}$ and fixed guessing set size of 10^9.

Metric	$\tau_{\max}$ = 1	$\tau_{\max}$ = 2	$\tau_{\max}$ = 5	$\tau_{\max}$ = 10
Unique passwords	170,609,130	262,456,443	328,845,663	357,299,090
N_{match}	836	1030	1037	1203
Cracking rate (over PassGAN [27])	17.06% 15.2%	21.28% 43.7%	26.48% 78.8%	24.81% 67.5%

Table 5: Number of passwords with different lengths in the RockYou test set.

Test set	Length = 6	Length = 7	Length = 8	Length = 9	Length = 10
RockYou	198,582	250,442	301,813	243,396	202,641

Table 6: Cracking rates of PassGAN [27] and proposed G-Pass under different password lengths.

Model	Length = 6 (%)	Length = 7 (%)	Length = 8 (%)	Length = 9 (%)	Length = 10 (%)
PassGAN [27]	29.96	22.05	13.28	6.62	3.15
G-pass (τ_{max} = 1)	33.68	24.50	15.82	8.36	3.87
Improvement	3.72	2.45	2.54	1.74	0.72
Relative improvement	12.4	11.1	19.1	26.2	22.8
G-pass (τ_{max} = 2)	38.32	29.83	20.34	12.18	6.36
Improvement	8.36	7.78	7.06	5.56	3.21
Relative improvement	27.9	35.2	53.1	83.9	101.9
G-pass (τ_{max} = 5)	45.06	36.64	25.67	16.17	9.34
Improvement	15.10	14.59	12.39	9.55	6.19
Relative improvement	50.4	66.1	93.3	144.2	196.5
G-pass (τ_{max} = 10)	43.15	34.52	23.95	14.74	8.24
Improvement	13.19	12.47	10.67	8.12	5.09
Relative improvement	44.0	56.5	80.3	122.6	161.5

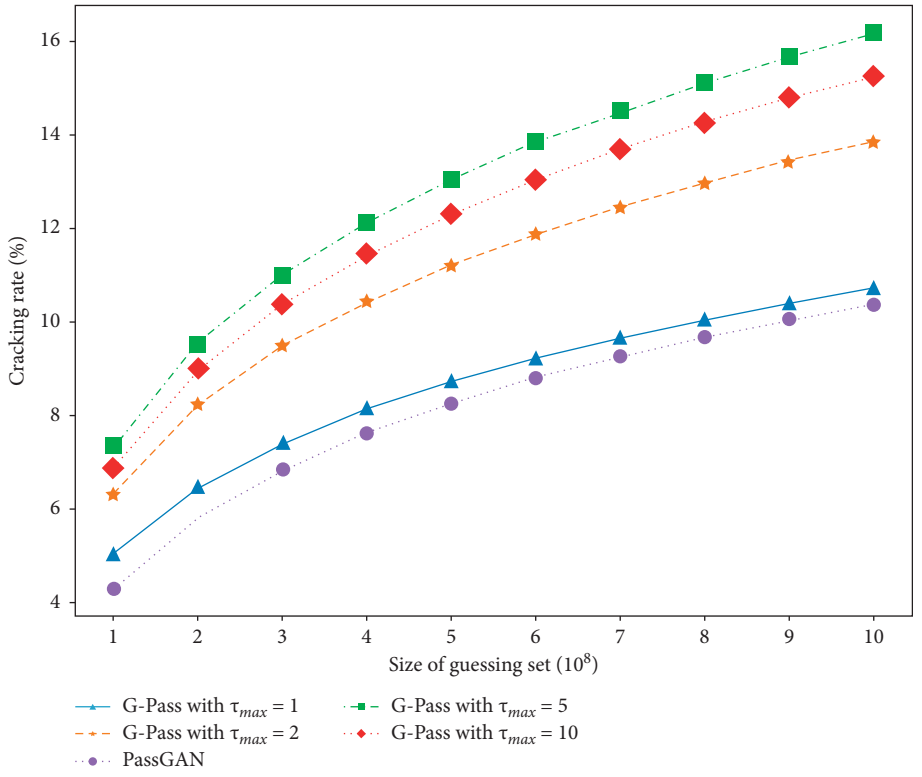

Figure 7: Cracking rate of PassGAN and proposed G-Pass on the LinkedIn test set with respect to τ_{max} and size of guessing set.

Table 7: Cracking rates of PassGAN [27] and proposed G-Pass on two testing sets with a guessing set size of 10^9.

Model	RockYou testing set (%)	LinkedIn testing set
PassGAN [27]	14.81	10.36%
G-pass(τ_{max} = 1)	17.06	10.72% (3.5% improvement)
G-pass(τ_{max} = 2)	21.28	13.84% (33.6% improvement)
G-pass(τ_{max} = 5)	26.48	16.17% (56.1% improvement)
G-pass(τ_{max} = 10)	24.81	15.26% (47.3% improvement)

belong to different fields. RockYou is a social network service website and LinkedIn is a job-search website. Since their users differ in many aspects, there is a difference in the distribution of passwords.

Compared to the performance on the LinkedIn test set, a 6%–10% increase in the cracking rate can be obtained with G-Pass on the RockYou test set. In contrast, only a 4% increase in the cracking rate can be achieved with PassGAN on the same test set. This phenomenon can be explained in terms of sample quality. N_{match} of G-Pass on the RockYou test set is lower than that of PassGAN, suggesting that G-Pass simulates the distribution of the RockYou dataset better. Therefore, the gap in cracking rate between G-Pass and PassGAN tends to shrink when they are applied on other test sets. Nevertheless, G-Pass still outperforms PassGAN on the LinkedIn test set.

6. Concluding Remark

In this paper, a novel password guessing model named G-Pass has been proposed, which combines the Gumbel-Softmax and GAN so that better performance can be achieved. In the proposed G-Pass, the LSTM is used to construct the generator to get higher quality samples, while convolutional layers with different filter sizes are designed to capture comprehensive information to guide the update of the generator. Furthermore, a trade-off between sample diversity and quality can be obtained with the proposed G-Pass by controlling the temperature parameter during training. In our future work, we will integrate the traditional password guessing method with G-Pass to improve the password cracking ability on different test sets.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the Guangdong Province Key Area Research and Development Program of China (2019B010137004), the National Natural Science Foundation of China (61772208), and the Natural Science Foundation of Guangdong Province of China (2021A1515011798).

References

[1] J. Wang, H. Han, H. Li, S. He, P. K. Sharma, and L. Chen, "Multiple strategies differential privacy on sparse tensor factorization for network traffic analysis in 5G," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1939–1948, 2022.

[2] J. Zhang, S. Zhong, T. Wang, H. Chao, and J. Wang, "Blockchain-based systems and applications: a survey," *Journal of Internet Technology*, vol. 21, no. 1, pp. 1–14, 2020.

[3] J. Wang, W. Chen, L. Wang, R. Simon Sherratt, O. Alfarraj, and A. Tolba, "Data secure storage mechanism of sensor networks based on blockchain," *Computers, Materials & Continua*, vol. 65, no. 3, pp. 2365–2384, 2020.

[4] G. J. Ra, C. H. Roh, and I. Y. Lee, "A key recovery system based on password-protected secret sharing in a permissioned blockchain," *Computers, Materials & Continua*, vol. 65, no. 1, pp. 153–170, 2020.

[5] H. T. Wu, Y. Zheng, B. Zhao, and J. Hu, "An anonymous reputation management system for mobile crowdsensing based on dual blockchain," *IEEE Internet of Things Journal*, p. 1, 2021.

[6] Z. Li, J. Wang, C. Choi, and W. Zhang, "Multi-factor password-authenticated key exchange via Pythia PRF service," *Computers, Materials & Continua*, vol. 63, no. 2, pp. 663–674, 2020.

[7] B. Zahednejad, L. Ke, and J. Li, "A novel machine learning-based approach for security analysis of authentication and key agreement protocols," *Security and Communication Networks*, vol. 2020, Article ID 8848389, 15 pages, 2020.

[8] S. Ju, H. Seo, S. Han, J. c. Ryou, and J. Kwak, "A study on user authentication methodology using numeric password and fingerprint biometric information," *BioMed research international*, vol. 2013, Article ID 427542, 17 pages, 2013.

[9] C. Herley and P. Van Oorschot, "A research agenda acknowledging the persistence of passwords," *IEEE Security & privacy*, vol. 10, no. 1, pp. 28–36, 2011.

[10] Z. Zheng, H. Cheng, Z. Zhang, Y. Zhao, and P. Wang, "An alternative method for understanding user-chosen passwords," *Security And Communication Networks*, vol. 2018, Article ID 6160125, 12 pages, 2018.

[11] Y. Li, H. Wang, and K. Sun, "A study of personal information in human-chosen passwords and its security implications," in *Proceedings of the IEEE INFOCOM The 35th Annual IEEE International Conference on Computer Communications*, vol. 07, pp. 1–9, San Francisco, CA, USA, April 2016.

[12] W. Zhang, K. Li, T. Li, S. Niu, and Z. Gao, "Cm-droid: secure container for android password misuse vulnerability," *Computers, Materials & Continua*, vol. 59, no. 1, pp. 181–198, 2019.

[13] M. Dell'Amico, P. Michiardi, and Y. Roudier, "Password strength: an empirical analysis," in *Proceedings of the 29th Conference on Information Communications*, pp. 983–991, Diego, CA, USA, May 2010.

[14] S. A. Alsuhibany, "A camouflage text-based password approach for mobile devices against shoulder-surfing attack," *Security and Communication Networks*, vol. 2021, Article ID 6653076, 11 pages, 2021.

[15] J. Ma, W. Yang, M. Luo, and N. L, "A study of probabilistic password models," in *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, pp. 689–704, Berkeley, CA, USA, May 2014.

[16] M. Weir, S. Aggarwal, B. D. Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *Proceedings of the 2009 30th IEEE Symposium On Security And Privacy*, pp. 391–405, Oakland, CA, USA, May 2009.

[17] K. Fu, J. Peng, H. Zhang, X. Wang, and F. Jiang, "Image super-resolution based on generative adversarial networks: a brief review," *Computers, Materials & Continua*, vol. 64, no. 3, pp. 1977–1997, 2020.

[18] W. Nie, N. Narodytska, and A. Patel, "Relgan: relational generative adversarial networks for text generation," in *Proceedings of the International Conference on Learning*

Representations, pp. 1–20, Vancouver, Canada, September 2018.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, vol. 02, pp. 2672–2680, Montreal Canada, December 2014.

[20] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung, "MGAN: training generative adversarial nets with multiple generators," in *Proceedings of the International Conference on Learning Representations*, pp. 1–24, New Orleans, LA, USA, February 2018.

[21] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," *International conference on machine learning*, vol. 70, pp. 214–223, 2017.

[22] I. Gulrajani, F. Ahmed, M. Arjovsky, I. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," in *Proceedings of the 31stInternational Conference On Neural Information Processing Systems*, pp. 5769–5779, Long Beach, CA, USA, December 2017.

[23] Z. Xia, P. Yi, Y. Liu, B. Jiang, W. Wang, and T. Zhu, "GENPass: a multi-source deep learning model for password guessing," *IEEE Transactions on Multimedia*, vol. 22, no. 5, pp. 1323–1332, 2020.

[24] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proceedings of the 12th ACM Conference on Computer and Communications Security*, pp. 364–372, Raleigh, NC, USA, 2005.

[25] W. Melicher, B. Ur, S. M. Segreti et al., "Fast, lean, and accurate: modeling password guessability using neural networks," in *Proceedings of the 25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 175–191, Vancouver, Canada, 2016.

[26] L. Xu, C. Ge, W. Qiu et al., "Password guessing based on LSTM recurrent neural networks," in *Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, vol. 1, pp. 785–788, Guangzhou, China, July 2017.

[27] B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz, "Passgan: a deep learning approach for password guessing," *Applied Cryptography and Network Security*, vol. 11464, pp. 217–237, 2019.

[28] S. Nam, S. Jeon, H. Kim, and J. Moon, "Recurrent gans password cracker for iot password security enhancement," *Sensors*, vol. 20, no. 11, 2020.

[29] D. Pasquini, A. Gangwal, G. Ateniese, M. Bernaschi, and M. Conti, "Improving Password Guessing via Representation learning," *IEEE Symposium on Security and Privacy (SP)*, pp. 1382–1399, 2021.

[30] T. D. Nguyen, T. Le, H. Vu, and D. Phung, "Dual discriminator generative adversarial nets," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 2667–2677, Long Beach, NY, USA, December 2017.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] P. Zhou, W. Shi, J. Tian et al., "Attention-based bidirectional long short-term memory networks for relation classification," *Proceedings of the 54th annual meeting of the association for computational linguistics*, vol. 02, pp. 207–212, 2016.

[33] T.-T.-H. Le, S. Heo, and H. Kim, "Toward load identification based on the hilbert transform and sequence to sequence long short-term memory," *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3252–3264, July 2021.

[34] H. Wang, Z. Qin, and T. Wan, "Text generation based on generative adversarial nets with latent variables," *Pacific-asia Conference on Knowledge Discovery and Data Mining*, vol. 10938, pp. 92–103, 2018.

[35] Y. Zhang, Z. Gan, and L. Carin, "Generating text via adversarial training," *NIPS workshop on Adversarial Training*, vol. 21, pp. 21–32, 2016.

[36] M. J. Kusner and J. M. H. Lobato, "Gans for sequences of discrete elements with the gumbel-softmax distribution," pp. 1611–1616, 2016, http://arxiv.org/abs/1611.04051.

[37] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: a continuous relaxation of discrete random variables," in *Proceedings of the International Conference on Learning Representations*, pp. 1–20, Toulon, France, April 2017.

[38] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *Stat*, vol. 1050, pp. 5–17, 2017.

[39] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," *ICLR (Poster)*, pp. 1–15, 2015.