Computers, Materials & Continua
DOI:10.32604/cmc.2021.017950
Article

Tech Science Press

GAN-GLS: Generative Lyric Steganography Based on Generative Adversarial Networks

Cuilin Wang[1], Yuling Liu[1,*], Yongju Tong[1] and Jingwen Wang[2]

[1]College of Computer Science and Electronic Engineering, Hunan University, Changsha, 410082, China
[2]Department of Computer Science, Elizabethtown College, PA, 17022, USA
[*]Corresponding Author: Yuling Liu. Email: yuling_liu@hnu.edu.cn
Received: 18 February 2021; Accepted: 13 April 2021

Abstract: Steganography based on generative adversarial networks (GANs) has become a hot topic among researchers. Due to GANs being unsuitable for text fields with discrete characteristics, researchers have proposed GAN-based steganography methods that are less dependent on text. In this paper, we propose a new method of generative lyrics steganography based on GANs, called GAN-GLS. The proposed method uses the GAN model and the large-scale lyrics corpus to construct and train a lyrics generator. In this method, the GAN uses a previously generated line of a lyric as the input sentence in order to generate the next line of the lyric. Using a strategy based on the penalty mechanism in training, the GAN model generates non-repetitive and diverse lyrics. The secret information is then processed according to the data characteristics of the generated lyrics in order to hide information. Unlike other text generation-based linguistic steganographic methods, our method changes the way that multiple generated candidate items are selected as the candidate groups in order to encode the conditional probability distribution. The experimental results demonstrate that our method can generate high-quality lyrics as stego-texts. Moreover, compared with other similar methods, the proposed method achieves good performance in terms of imperceptibility, embedding rate, effectiveness, extraction success rate and security.

Keywords: Text steganography; generative adversarial networks; text generation; generated lyric

1 Introduction

Steganography is an important research area in the field of network security. For many years, many efforts have been made to embed secret information into some public carriers, such as images, audios, and texts. Because there is relatively less redundant encoding space in texts than in other carriers, research on text steganography is a greater challenge. Conventional text steganographic methods hide information by modifying the format or content of the text carriers. However, due to the smaller amount of information redundancy, the embedding capacity of text steganography methods based on modifying the format is limited. In addition, with the development of steganalysis algorithms, the security of these methods cannot be guaranteed,

because this kind of method cannot avoid the detection and attack of various steganalysis algorithms. Based on these disadvantages, some scholars have put forward the concept of coverless steganography [1,2], a method that, rather than making any changes to the carriers, uses retrieval and generation technologies to obtain some carriers that can express secret information.

The coverless steganography method based on text retrieval is driven by secret information, where the existing features of the retrieved texts can represent secret information. Chen et al. [3] proposed the first coverless text steganography method, which divided a Chinese character into several combinations of Chinese character parts by a Chinese character mathematical expression, then used these Chinese character parts as the secret labels to retrieve the combination of secret labels and secret information from the text corpus, and finally obtained the stego-text. However, this method could only embed one keyword in one text. To improve the embedding rate of a single text, Liu et al. [4] proposed a method that used the Chinese Pinyin as the secret labels to complete the hiding of information. They embedded the secret information by mapping the part of speech (POS) of keywords to the numbers. Long et al. [5] proposed a coverless steganography method by retrieving web pages. This method was driven by secret information and searched the stego-text in the web pages. Through constructing the word vector model by use of the Word2Vec tool, the synonyms were replaced when the keyword failed to be searched, thereby improving the embedding and extraction success rates. Although such retrieval methods can avoid the detection and attacks of steganalysis algorithms, they need to retrieve a large number of texts in order to embed a small amount of secret information. This means that the information embedding dimension is based on the whole text, and the embedding rate is low. To solve this problem, some scholars have put forward the concept of coverless steganography based on text generation.

Through deep learning, natural language processing, and text generation technologies, new text can be generated according to the secret information, which can improve the embedding rate while hiding the secret information. Zhang et al. [6] used the encoder-decoder model to generate Chinese Tang poetry as stego-text. They compressed the previously generated Tang poetry into an input vector to guide the generation of the next Tang poetry, which was similar to the task of machine translation. Wang et al. [7,8] proposed a method to generate Tang poetry with a consistent theme using a recurrent neural network (RNN) with an attention mechanism. In the process of training and generating Tang poetry, this method focused on the important parts of each poem and the sentences with strong dependence, thereby improving the thematic consistency of all the Tang poems. To embed the secret information, Luo et al. [9,10] used a Markov chain to generate Song Ci and a long short-term memory (LSTM) model to generate Tang poetry, and embedded the secret information in the process of text generation. However, in today's culture, most people use texts for communication, and therefore the transmission of ancient poetry on the network, requiring a special type of text, will easily arouse the suspicion of the attackers. To solve this problem, Tong et al. proposed a method of using LSTM to generate the popular lyrics and hide information, which can not only generate the natural lyrics but also embed secret information [11]. Unlike the generative text steganography of special genres, such as Tang poetry, Song Ci, and lyrics, researchers have proposed some general text steganography methods based on text auto-generation technology. Fang et al. [12] proposed a text steganographic method with the LSTM network, which was the first work to use LSTM to learn the statistical language model of natural text. Subsequently, Yang et al. [13] changed the encoding process to dynamic coding based on conditional probability distribution and proposed two encoding modes: fixed-length coding (FLC) and variable-length coding (VLC). Kang et al. [14] proposed a generative text steganography based on an LSTM network and the attention mechanism with the keywords.

Because previous works focused only on generating steganographic sentences with better quality, Yang et al. [15] proposed a linguistic steganography method based on a variational autoencoder (VAE), which can guarantee perceptual-imperceptibility and statistical-imperceptibility. To improve the speed and embedding capacity, Xiang et al. [16] proposed a linguistic steganography method based on character-level text generation.

Although the methods based on text generation can improve the embedding rate, these methods are all based on the selection of multiple candidates with the highest probability, and the binary coding of these candidate items in the process of hiding information. In these methods, the quality of the generated text is affected by the secret information, and because there are many redundant candidate items in the candidate groups, the generated candidate group files will require a large amount of extra disk space.

To solve this problem, we propose a new method of coverless text steganography that applies a generative adversarial network to the task of generating lyrics. The method is called GAN-GLS. According to the characteristics of training data, we propose a method of hiding information based on invisible characters at the end of each lyric line. Experiments show not only that the GAN-GLS can further improve the embedding rate but also that the embedding rate is no longer limited by the length of the lyrics, providing the method with better scalability of the embedding rate.

Our contributions are as follows. First, we applied a GAN to the task of generating lyrics, and then hid the secret information. Second, we proposed a new method of hiding information based on the data characteristics of a GAN.

The rest of the paper is organized as follows. In Section 2, we introduce the concept of the GAN and its application in the text generation. In Section 3, we describe the proposed method. In Section 4, we evaluate the proposed method. Finally, we offer our conclusions in Section 5.

2 Preliminaries

In this section, we introduce the concept and the application of GANs, then describe how we use them in our experiments.

2.1 Generative Adversarial Networks

In 2014, Goodfellow proposed the GAN, a powerful generative model [17] whose main purpose is to train the generator G and the discriminator D by playing a minimax game. The generator is responsible for generating composite data, and the discriminator is responsible for judging whether the input data is the real sample or the composite data. The main goal of the generator is to generate data that can confuse the discriminator as much as possible in order to make its judgment wrong, while the goal of the discriminator is to identify the authentic data as much as possible. The optimization function of the entire training process and the optimization trend of data distribution are

$$\min_{G}\max_{D} V(D, G) = E_{x\sim p_{data}(x)}[\log D(X)] + E_{z\sim p_{z(z)}}[\log(1 - D(G(z)))] \tag{1}$$

2.2 Application of GANs

Due to their powerful generative capacity, GANs have been successfully applied to many research fields, such as computer vision, image synthesis, and natural language processing. Scholars have proposed many steganography methods using GANs [18,19]. GAN-based steganography

consists of three categories: cover modification methods based on GANs, cover selection methods based on GANs, and cover synthesis methods based on GANs. Reference [20] reviewed recent advances in image steganography with GANs, but did not mention text steganography.

Because the training samples of the GANs are continuous data, traditional GANs are not suitable for the text field with discrete characteristics. Therefore, some researchers have concentrated on how to modify the basic model to fit the text data. Yu et al. [21] proposed Seq-GAN, which is a milestone for the application of GANs to text generation. The authors added reinforcement learning [22] to the GANs, regarded text generation as a sequential decision-making problem, and fed back the information of the generated sentences through the policy gradient in reinforcement learning, thereby guiding the generator to update the parameters. At the same time, in order to evaluate whether the token generated in the short term is reasonable in the long term, in this paper we used Monte Carlo [23] to compute the result of the generated token.

Yang et al. [24] proposed a text steganography method based on GANs (GAN-TStega), which is the first method to use GANs for text steganography. However, they also embedded the secret information by encoding the conditional probability distribution of each word, similar to other text generation-based steganography methods. In this paper, we use GANs to generate the lyrics as the stego-text, but change the manner of embedding to one that embeds the secret information by calculating the number of invisible characters of the generated lyrics.

3 Method

In this section, we describe the proposed method (GAN-GLS). Fig. 1 shows that the proposed framework consists of two phases. It can be seen that the candidate words based on the conditional probability distribution are not used in the embedding of secret information.

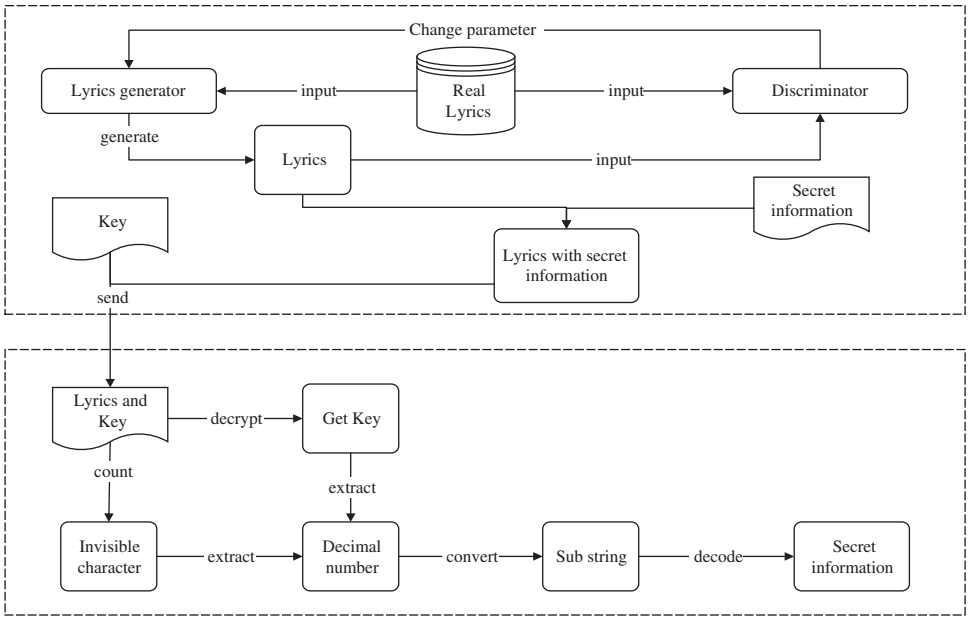

Figure 1: Flowchart of the proposed method

In the first phase, we train the GAN-based generating lyrics model on a lyrics dataset that is constructed by collecting a large number of lyrics with different styles and different themes from the NetEase cloud music streaming service. When the model converges, we obtain the lyrics generator (G network).

In the second phase, the sender combines the trained G network with the initial sentence to generate a line of lyric, counts the invisible characters of the generated lyrics, and uses this value to process the binary string of secret information. The sender then transmits the generated lyrics and key to the receiver. The receiver extracts the secret information after receiving the lyrics file and key. Tab. 1 presents the notations used in this paper.

Table 1: Notations used in this paper

Symbol	Definition
Len	Maximum length of lyrics
m	Number of generated words
n	Number of generated space characters (zero)
Info	Secret information binary string
Key	Key for secret information extraction
Num	Decimal number converted from binary sub-string

3.1 Generating Lyrics Model Based on GANs

In the process of lyrics generation, we take advantage of GANs' powerful ability in adversarial learning to generate realistic lyrics. The model contains two networks: a generator network (G) and a discriminator network (D). In order to fool D, the weight of G is updated, and at the same time, the D's weight is updated by distinguishing between the fake and real lyrics.

3.1.1 Generator

To prevent the gradient explosion and disappearance, we chose the LSTM model, which is a type of RNN as our generator. The probability sampling or the value with the highest probability is directly selected as the next word of the lyric sequence, and the iteration can complete the generation of the entire sequence.

3.1.2 Discriminator

The main purpose of the discriminator is to judge whether the input text is a real lyric sentence or a machine-generated lyric sentence. It plays the role of Eve in the covert communication system. While generating the lyrics model, we use the text CNN model as the discriminator. For the discriminator, we use the highway net model, as shown below:

$$\tau = \sigma\ (W_T \overline{c} + \mathrm{b}_T) \tag{2}$$

$$\overline{C} = \tau\, H\ (\overline{c},\ W_H) + (1 - \tau)\, \overline{\mathrm{c}}$$

where $\overline{\mathrm{c}}$ represents the word vector, $1 - \tau$ represents the inflow proportion of the original information, and W_T, W_H, $\mathrm{b_T}$ is the training weight. Then we use the full connection layer and the activation function to form the final binary classification probability. At the same time, we add the dropout mechanism to the discriminator network to prevent overfitting.

3.1.3 Update Strategy

The generation model is defined as $G_\theta(\mathrm{y}_t \mid Y_{1:t-1})$, θ is the model parameter, y_t is the output at time t, and $Y_{1:\mathrm{t}-1}$ is the output before time t. $Q_{D_\phi}^{G_\theta}(s, a)$ represents the behavior value function of the sequence. s is the generated tokens, and a is the next token to be generated. For the entire sequence, this value function can be defined as:

$$Q_{D_\phi}^{G_\theta}(\mathrm{a} = y_T, s = Y_{1:T-1}) = D(Y_{1:T}) \tag{3}$$

where $D_\phi(Y_{1:T})$ represents the reward input from the entire sequence to the output of the discriminator D_ϕ. For the T-time series, if the reward is evaluated, the Monte Carlo tree algorithm is used to generate the last T-t tokens using the generator G_β. We use K times, which is expressed as follows:

$$\left\{ Y_{1T}^1, \ldots, Y_{1:T}^1 \right\} = MC^{G_\beta}(Y_{1:t}; K) \tag{4}$$

For each sampling, the complete sequence generated by the discriminator is also rewarded, and for k samples, the expectation is taken. Accordingly, there are

$$Q_{D\phi}^{G_\theta}(\mathrm{s} = Y_{1:t-1}, a = y_t) = \begin{cases} D_\phi(Y_{1:t}) & t = T \\ \dfrac{1}{K} \displaystyle\sum_{\mathrm{k}=1}^{K} D_\phi\left(Y_{1:T}^k\right), \; Y_{1:T}^k \in MC^{G_\beta}(Y_{1:t}; K) & t < T \end{cases} \tag{5}$$

Then, for time t, the value function is:

$$V^{G_\theta}(s = Y_{1:\mathrm{t}-1}) = \sum_{y_t \in Y} G_\theta(y_t \mid Y_{1:t-1}) \, Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t) \tag{6}$$

The goal of the generation model is to maximize the entire generation sequence reward using the following formula:

$$J(\theta) = E[R_T \mid \mathrm{s}_0, \theta] = \sum_{y_1 \in Y} G_\theta(y_1 \mid s_0) \, Q_{D_\phi}^{G_\theta}(s_0, y_1) \tag{7}$$

The gradient is as follows:

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \left[\sum_{y_1 \in Y} G_\theta(y_1 \mid s_0) \, Q_{D_\phi}^{G_\theta}(s_0, y_1) \right] \\ &= \sum_{y_1 \in Y} \left[\nabla_\theta G_\theta(y_1 \mid s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1) + G_\theta(y_1 \mid s_0) \cdot \nabla_\theta Q_{D_\phi}^{G_\theta}(s_0, y_1) \right] \\ &= \sum_{y_1 \in Y} \left[\nabla_\theta G_\theta(y_1 \mid s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1) + G_\theta(y_1 \mid s_0) \cdot \nabla_\theta Q_{D\phi}^{G_\theta}(Y_{1:1}) \right] \\ &= \sum_{y_1 \in Y} \nabla_\theta G_\theta(y_1 \mid s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1) \end{aligned}$$

$$
\begin{aligned}
&+\sum_{y_1\in Y} G_\theta\left(y_1 \mid s_0\right)\left[\sum_{y_2\in Y} \nabla_\theta G_\theta\left(y_2 \mid Y_{1:1}\right)\cdot Q_{D\phi}^{G_\theta}\left(Y_{1:1} \mid y_2\right)+G_\theta\left(y_2 \mid Y_{1:1}\right)\cdot \nabla_\theta Q_{D\phi}^{G_\theta}\left(Y_{1:1} \mid y_2\right)\right]\\
&=\sum_{y_1\in Y} \nabla_\theta G_\theta\left(y_1 \mid s_0\right)\cdot Q_{D_\phi}^{G_\theta}\left(s_0, y_1\right)+\sum_{y_{1:1}} P\left(Y_{1:1} \mid s_0; G_\theta\right)\sum_{y_2\in Y} \nabla_\theta G_\theta\left(y_2 \mid Y_{1:1}\right)\cdot Q_{D\phi}^{G_\theta}\left(Y_{1:1} \mid y_2\right)\\
&+\sum_{Y_{1:2}} P\left(Y_{1:2} \mid s_0; G_\theta\right)\nabla_\theta V^{G_\theta}\left(Y_{1:2}\right)\\
&=\sum_{\mathrm{t}=1}^{T}\sum_{Y_{1:t-1}} P\left(Y_{1:t-1} \mid s_0; G_\theta\right)\sum_{y_t\in Y} \nabla_\theta G_\theta\left(y_t \mid Y_{1:t-1}\right)\cdot Q_{D\phi}^{G_\theta}\left(Y_{1:t-1} \mid y_t\right)\\
&=\sum_{\mathrm{t}=1}^{T} E_{Y_{1:t-1}\sim G_\theta}\left[\sum_{y_t\in Y} \nabla_\theta G_\theta\left(y_t \mid Y_{1:t-1}\right)\cdot Q_{D\phi}^{G_\theta}\left(Y_{1:t-1} \mid y_t\right)\right]
\end{aligned}
\tag{8}
$$

Therefore, the parameters of the generation model are updated to $\theta=\theta+\mathrm{a}_h\nabla_\theta J\left(\theta\right)$.

The goal of the discrimination model is to improve the probability of judging real data and reduce the probability of generating data. This value function can be defined as:

$$
\min -E_{Y\sim p_{data}}\left[\log D\phi\left(Y\right)\right]-E_{Y\sim G_\theta}\left[\log\left(1-D\phi\left(Y\right)\right)\right] \tag{9}
$$

When the probability of judging the real data is greater, $-E_{Y\sim\mathrm{data}}\left[\log D\phi\left(Y\right)\right]$ is smaller. The smaller the probability of generating data, the smaller $-E_{Y\sim\mathrm{G}_\theta}\left[\log\left(1-D\phi\left(Y\right)\right)\right]$. Using the loss function, the discrimination model is trained in reverse.

3.1.4 Penalty Mechanism

The model for text generation proposed by Yu et al. [21] is a GAN based on policy gradient, and the information that the discriminator feeds back to the generator is the reward value for the authenticity judgment of input data. After receiving the reward value, the generator can learn which data are closer to the real data distribution and generate those data with the higher reward value next time to optimize the generated results. However, this will lead to generating repetitive results, meaning that the model will discard the diversity of generated results to ensure the high quality of the generated results.

To solve this problem, we use the GANs based on the penalty value. The structure diagram is shown in Fig. 2.

The difference between this and Seq-GAN is that the information that the discriminator feeds back to the generator is no longer the reward value of the input data, but the penalty value. The relationship between the two values is $\mathrm{r}=1-p$, where r is the reward value and p is the penalty value. The reason for this adjustment is that after receiving the feedback of the penalty value, the generator will know which generated data are not quality data. Then, while generating data the next time, it will avoid generating the same data again and try to generate other data, thereby achieving diversity of the generated results.

Based on the reward mechanism, the discriminator in the GAN regards the probability that the input sentence is judged to be true as the reward, while the generator updates its gradient by maximizing the reward value, which will lead to mode collapse in the training process. As shown

in Fig. 3, the result is that the generated data is repeated after multiple iterations of the training process.

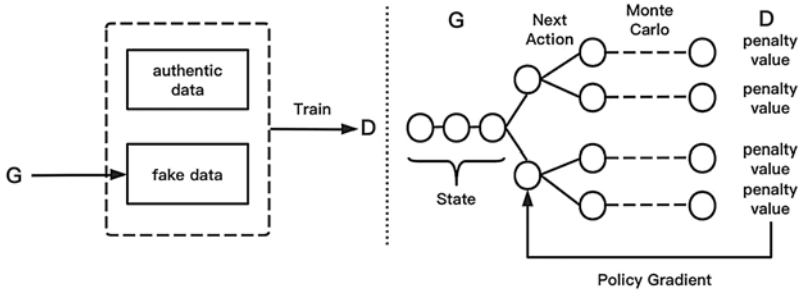

Figure 2: GAN based on the penalty mechanism

```
173 569 1920 113 55 3
173 569 1920 113 55 3
173 569 1920 113 55 3
173 569 1920 113 55 3
173 569 1920 113 55 3
```

Figure 3: Generated results based on reward mechanism

The GAN used in this paper is based on the penalty mechanism, and the optimization function of the generator is

$$J_G(X) = E_X \sim_{Pg} \left[G\left(X \mid S; \theta_g\right) V(X; \theta_d) \right] \tag{10}$$

The discriminator feeds back the false probability of lyrics as punishment to the generator, and the generator updates its gradient by minimizing the value of punishment, which not only ensures the confrontation training of the two sub-models but also avoids the occurrence of the mode collapse phenomenon. Because there is an $r = 1 - p$ relationship between the penalty value and the reward value, according to the characteristics of the GAN, this will make the generator prefer to generate a line lyric with less probability and thereby avoid generating many repetitive but more probability lyrics, as shown in Tab. 2, where the left side is the generated lyrics and the right side is the corresponding numerical index in the text dictionary.

Furthermore, the training of the GAN can be seen as the process of playing zero games between two sub-models. The purpose of the GAN is to confuse the data generated by the generator with the discriminator and therefore cannot distinguish the authenticity of the generated text. As a result, when the discriminator judges the authenticity of the generated text at an intermediate value, it is the best state of the model. As shown in Fig. 4, after 200 times of training, the discriminator's accuracy gradually tends toward the intermediate value of 0.5, which is the best state of the game between the generator and the discriminator. If the accuracy of the discriminator is too high, it means that the output result of the generator is not enough to confuse the discriminator. If the accuracy of the discriminator is too low, it means that the ability of the discriminator to distinguish the true and false data is too low. Only when the accuracy

is about 0.5, the data generated by the generator making the discriminator unable to distinguish, and the training goal of the GAN is achieved.

Table 2: Generated lyrics based on penalty value and corresponding indices

Generated lyric	Corresponding index
We are friends	14 56 220 0 0 0 0 0 0 0 0 0 0 0 0
To have you is to have whole world	5 78 2 29 5 78 354 134 0 0 0 0 0 0 0
Mom loves me again	1293 815 4 114 0 0 0 0 0 0 0 0 0 0 0
Happy is the most important thing	440 29 3 652 3214 195 0 0 0 0 0 0 0 0 0
Even if you have someone else	146 44 2 78 262 296 0 0 0 0 0 0 0 0 0
The story is like a dream	3 678 29 20 7 355 0 0 0 0 0 0 0 0 0
Life may be a kiss	96 508 22 7 298 0 0 0 0 0 0 0 0 0 0
Walking alone in a lonely street	569 153 10 7 313 736 0 0 0 0 0 0 0 0 0
I love you	1 19 2 0 0 0 0 0 0 0 0 0 0 0 0

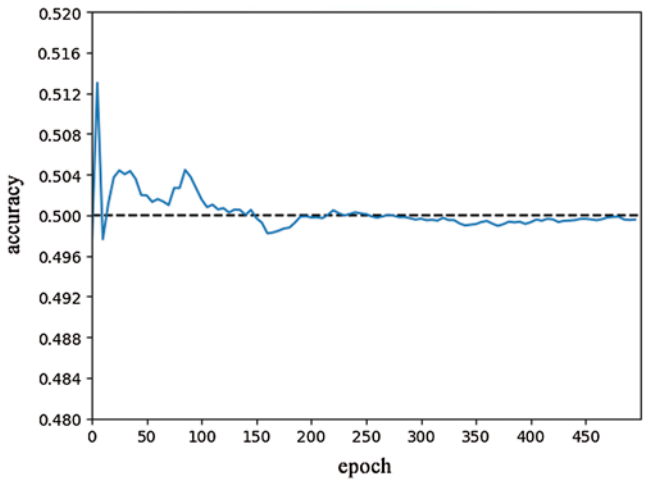

Figure 4: Accuracy of the discriminator

3.2 Information Hiding Algorithm

Once the GAN-based generating lyrics model is trained, the sender follows the algorithm below for information hiding. When the secret information is hidden, the *key* is obtained by counting the invisible characters of the generated lyrics and using this value to process the binary string of secret information. The steps are:

Step 1: Input the secret information *Info*. Each letter of secret information is converted into an 8-bit binary string, and then all the binary strings are spliced together. Input the maximum length of lyrics *len*.

Step 2: Generate a line of lyric using the trained model and an initial sentence.

Step 3: Calculate the number of invisible characters of the generated lyric, which is denoted as n, where $n = len - m$.

Step 4: Select a binary sub-string g of length n from the binary string of secret information, and convert the binary substring to a decimal number.

$$num = conversion(substring) \tag{11}$$

Step 5: Multiply num and n as $sub - key$, and use the result value as part of the key. Then repeat the above steps until all secret information binary strings are processed.

Step 6: Link all sub-keys using "-" as the separator between each sub-key to obtain the final key, which is encrypted using any encryption method. After the lyrics are generated and the information embedded, the sender sends the generated lyrics file and key to the receiver.

3.3 Information Extraction Algorithm

After receiving the lyrics file and key, the receiver extracts the secret information according to the following steps.

Step 1: Use the decryption method agreed upon by the sender to decrypt the key.

Step 2: Segment the decrypted key according to the character "-" to obtain a set of $sub - key$.

Step 3: Calculate n, which is the number of spaces at the end of each lyric, and select the $sub - key$ in order, divide it by n, and obtain num. Then repeat the above steps until all $sub - keys$ are processed.

Step 4: Convert all num into binary sub-strings, and finally combine all binary sub-strings into a complete binary string in order, which is the binary representation of secret information.

Step 5: Obtain the secret information using the decoding method.

4 Experiments and Results

We designed several experiments to verify the performance of the proposed method.

4.1 Data Preparing

Since we expect that the proposed method can automatically generate realistic lyrics, we need a large number of real lyrics for model training. Due to the lack of publicly available lyrics datasets, we crawl a large number of English lyrics by the Scrapy scraping and web-crawling tool from the NetEase cloud music as our datasets. The details of the lyrics datasets are shown in Tab. 3.

Table 3: Details of the lyrics datasets

Item	Value
Number of lyrics	143386
Number of words	1570893
Number of letters	6921943
Average length of lyrics	10.95569

Furthermore, since traditional GANs were not suitable for the generation of lyrics, we chose the GANs based on Policy Gradient and Monte Carlo algorithm in [21] as the basic model of the proposed method. By analyzing the data characteristic of the model, it can be found that the data fed into the GANs should be fixed length (just like the length and width of the image are fixed). However, the length of each line in the original lyric data is variable. Therefore, we needed to preprocess the dataset to meet the requirement of our model. Moreover, it can be found that the number of words in most lines of lyrics is about 10, and the maximum is 15. Therefore, we chose 15 as the maximum length of the lyrics ($len = 15$) in the experiment. For those lines whose length does not reach 15, we supplemented the space characters at the end (the space character corresponds to the number 0 in the constructed text dictionary), and the proposed hiding algorithm was based just on these invisible space characters at the end of such a line. The dataset format after preprocessing is shown in Fig. 5.

```
101 63 1240 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2472 4 1138 47 10 211 41 0 0 0 0 0 0 0 0 0 0 0 0
115 3 10 284 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
18 197 31 68 22 2349 0 0 0 0 0 0 0 0 0 0 0 0 0
1 203 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
654 654 139 136 1094 113 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 5: Dataset format after preprocessing

4.2 Evaluation Results and Discussion

Our experimental set-up consisted of TensorFlow v1.14.0, NumPy v1.16.0, and Keras v2.2.5. When training the discriminator, for the embedded layer, we used a 128-dimensional vector form to express each word and used a logarithmic loss function to optimize the model. At the same time, in the pre-training process, the batch size was set to 64, and the generator initialization learning rate was 0.01. During the confrontation training, the generator was trained once, and then the discriminator was trained once so that the generator and the discriminator were in balance.

Next, we conducted several experiments to verify the performance of the GAN-GLS model in terms of imperceptibility, embedding rate, effectiveness, extraction success rate, and security.

4.2.1 Imperceptibility Analysis

Imperceptibility is an important evaluation criterion in steganography, particularly for generative steganography. Unlike other text generation-based linguistic steganographic methods, which select multiple generated candidate items as the candidate groups to encode the conditional probability distribution, we hide the secret information based on the invisible characters at the end of each line lyric. In addition, the length of each line in the real lyric data is variable. Therefore, in this paper, we focus on the quality of the generated lyrics.

To show the distribution of generated lyrics and real lyrics, we first transformed a sequence into a word vector matrix through word vector embedding, and then obtained the word vector matrix of a word by calculating the average value of the words in the entire sentence. Finally, it was mapped to a two-dimensional space using the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm [25]. In this experiment, we selected 1000 training real lyrics texts and 200 generated lyrics texts (steganographic lyrics texts). The results are shown in Fig. 6, where the red dots are for training lyrics and the blue dots are for steganographic lyrics. It can be seen that the training lyrics text and steganographic lyrics texts are difficult to be separated in a two-

dimensional mapping space, which means that our method can generate high-quality lyrics and have good perceptual imperceptibility.

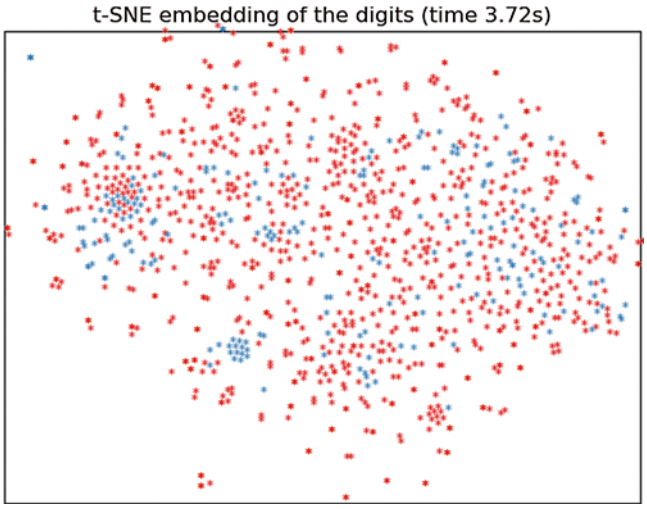

Figure 6: Comparison of the distribution of the training samples and the steganographic lyrics

4.2.2 Embedding Rate Analysis

Embedding rate (*ER*) is another important indicator for evaluating the performance of steganography. In this paper, embedding rate equals the ratio of the number of bits embedded in a lyric to the total number of words in the lyric itself. The formula is:

$$ER = \frac{\text{bit}}{total} \tag{12}$$

Using a specific example of lyrics generation and information hiding, we present the calculation process of embedding rate. First, we need to set three parameters:

(1) Length of lyrics: 60
(2) Topic sentence: I love you
(3) Secret information: see you later

Then we transform the topic sentences into the input vectors and then input them into the generator model. The secret information is converted into an 88-bits binary string. If the word number of the generated lyrics is 12, the generated lyrics are "if I were you, I would give up everything to you." The formula for embedding rate is 88 divided by 12, which equals 7.8. If the length of lyrics is set to 100, the embedding rate can reach 12.334. It can be found that the steganography method based on the invisible characters at the end of a line is affected by not only the length of the generated lyrics but also the number of invisible characters. The number of invisible characters is affected by the length of the longest lyrics. The greater the length of the longest lyrics, the more the invisible characters, and the higher the embedding rate.

Unlike other steganography methods based on the generative texts, the proposed method does not depend on the candidate groups in the process of information hiding and information extraction, but hides information by counting the number of space characters at the end of the

line of lyrics. Therefore, to some extent, the embedding rate of the proposed method is no longer limited to the total number of text characters. Compared with the method proposed by Tong et al. [11], the embedding rate is significantly higher than the candidate groups-based coverless steganography methods. Compared with the retrieval-based coverless steganography methods [3–5], the proposed method also is obviously superior, as shown in Tab. 4. Moreover, with the increase of the maximum length of lyrics, the embedding rate will increase.

Table 4: Comparison of embedding rate of different methods

Method	Embedding rate
Single keyword based search [3]	0.241
Multi-keywords based search [4]	0.572
Web-based retrieval [5]	2.543
Candidate group coding (size = 4) [11]	5.335
Candidate group coding (size = 8) [11]	6.88
Proposed method (60 words)	7.8
Proposed method (100 words)	12.334

4.2.3 Effectiveness Analysis

We conducted comparative experiments on time consumption and space consumption. The space consumption was mainly compared with the steganography method based on candidate group coding.

We measured the efficiency of the proposed method using the average time required to hide secret information of a specified length. We divided the experiment into four batches according to the capacity of secret information and calculated the average consumption time when the capacity of each batch of secret information is set to x bits. Experimental results showed that with the continuous increase of secret information capacity, the required time increases in proportion to the secret information capacity, as shown in Fig. 7. In exploring the reason for this, we found that the time spent was mainly used for model loading.

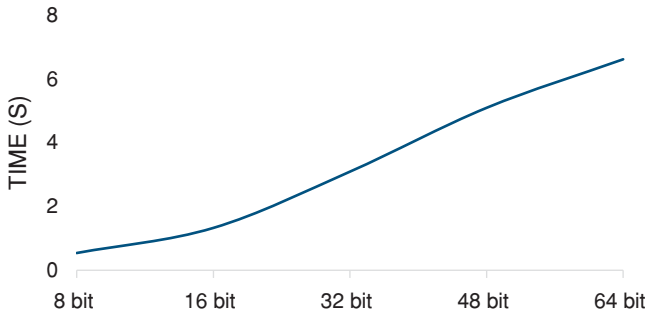

Figure 7: Average time needed to hide different secret information capacity

The steganography methods based on candidate group coding have to select multiple candidate tokens and carry out binary path coding to complete the hiding of information. In these

methods, in addition to the generated final token results, other redundant tokens will be selected to assist in the completion of the hiding of the secret information. Moreover, the generated candidate group also needs to be persisted into a specific one through some encoding files, which are used to transmit and assist in the extraction of secret information. However, the proposed method, based on the invisible characters at the end of a line, only needs to save the product of the number of invisible characters and secret information, without additional data. The encrypted key can be sent directly or persisted to a disk file, in which the disk space needed will be far less than that of the previous methods as shown in Tab. 5.

Table 5: Comparison of space consumption

Method	Space consumption (KB)
Candidate group coding (size = 4) [11]	1
Candidate group coding (size = 8) [11]	2
Proposed method (60 words)	1
Proposed method (100 words)	1

4.2.4 Analysis of Extraction Success Rate and Security

Some generative steganography methods have been proposed based on the GAN model, in which the generator directly generates the stego-carrier. However, because the stego-carrier generation depends on the optimization problem of the neural network [20], the analysis revealed that the extraction success rate of the existing methods cannot reach 100% when extracting secret information from the generated stego-carriers. In our experiment, we selected 100 representative lyrics from the generated 1000 lyrics to extract secret information. The experimental results show that the extraction success rate can reach 100%. Then we modified the selected 100 lyrics by substituting synonyms(such as changing "like" to "love"), re-adjusted the word order of lyrics, and then extracted the secret information again. The results show that the extraction success rate can still reach 100%, which confirms that the proposed method is robust. The reason is that the proposed GAN-GLS embeds the secret information by employing the data characteristic of the generation of GANs, rather than depending on the optimization during the generating process.

Security refers to the anti-steganalysis for a steganography method, which is another key criterion. GAN-GLS can generate modern lyrics texts of high naturalness, which are in line with people's daily communication habits. Therefore, the lyrics texts containing secret information are not likely to arouse the suspicion of the attackers in the transmission process, and the security can be guaranteed. At the same time, GAN-GLS is based on the invisible characters to hide secret information and does not need to make any changes to the lyrics texts. Theoretically, GAN-GLS can resist the detection of the latest steganalysis algorithms based on text generation, such as TS-CNN [26], TS-RNN [27], the text steganalysis method based on semantic analysis [28] proposed by Yang et al., and the CNN-based text steganalysis method proposed by Wen et al. [29].

5 Conclusions

We presented a novel coverless text steganography method that utilized GANs to generate lyrics. In the process of lyrics generation, we used the GANs based on the penalty mechanism as the basic model of the proposed method to generate the diversified lyrics. After generating

lyrics, we counted the number of invisible characters at the end of each line of lyrics according to the data characteristics of the GANs and used this value to process the binary string of secret information. Through this we achieved the embedding. Through experimental comparisons, we found that the proposed method can offer a higher embedding rate than other coverless text steganography methods and has better scalability of the embedding rate. In the future, we will try to add a steganalysis discriminator to assess the suitableness of the generated lyrics instead of steganalysis after the entire lyrics generation.

Funding Statement: This work was supported in part by the National Natural Science Foundation of China under Grant 61872134,61672222, author Y. L. Liu, http://www.nsfc.gov.cn/; in part by Science and Technology Development Center of the Ministry of Education under Grant 2019J01020, author Y. L. Liu, http://www.moe.gov.cn/; and in part by Science and Technology Project of Transport Department of Hunan Province under Grant 201935, author Y. L. Liu, http://jtt.hunan.gov.cn/; Science and Technology Program of Changsha City under Grant kh200519, kq2004021, author Y. L. Liu, http://kjj.changsha.gov.cn/.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Y. Luo, J. Qin, X. Xiang, Y. Tan and Z. He, "Coverless image steganography based on image segmentation," *Computers, Materials & Continua*, vol. 64, no. 2, pp. 1281–1295, 2020.

[2] Y. Tan, J. H. Qin, H. Tang, X. Y. Xiang, L. Tan *et al.,* "Privacy protection for medical images based on densenet and coverless steganography," *Computers, Materials & Continua*, vol. 64, no. 3, pp. 1797–1817, 2020.

[3] X. Y. Chen, H. Y. Sun, Y. Tobe, Z. L. Zhou and X. M. Sun, "Coverless information hiding method based on the Chinese mathematical expression," in *Proc. of Int. Conf. on Cloud Computing and Security*, Nanjing, China, pp. 133–143, 2015.

[4] Y. L. Liu, J. Wu and G. J. Xin, "Multi-keywords carrier-free text steganography method based on Chinese pinyin," *International Journal of Computational Science and Engineering*, vol. 21, no. 2, pp. 202–209, 2020.

[5] Y. Long, L. Y. Liu, Y. Q. Zhang, X. S. Ba and J. H. Qin, "Coverless information hiding method based on web text," *IEEE Access*, vol. 7, pp. 31926–31933, 2019.

[6] X. Zhang and M. Lapata, "Chinese poetry generation with recurrent neural networks," in *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, Doha, Qatar, pp. 670–680, 2014.

[7] Q. Wang, T. Luo, D. Wang and C. Xing, "Chinese song lambics generation with neural attention-based model," arXiv: 1604.06274, 2016.

[8] X. Yi, R. Li and M. Sun, "Generating Chinese classical poems with RNN encoder-decoder," in *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*. Guangzhou, China, pp. 211–223, 2017.

[9] Y. Luo, Y. Huang, F. Li and C. Chang, "Text steganography based on ci-poetry generation using Markov chain model," *KSII Transactions on Internet and Information Systems*, vol. 10, no. 9, pp. 4568–4584, 2016.

[10] Y. Luo and Y. Huang, "Text steganography with high embedding rate: Using recurrent neural networks to generate Chinese classic poetry," in *Proc. of the 5th ACM Workshop on Information Hiding and Multimedia Security*, New York, NY, USA, pp. 99–04, 2017.

[11] Y. J. Tong, Y. L. Liu, J. Wang and G. J. Xin, "Text steganography on RNN-generated lyrics," *Mathematical Biosciences and Engineering*, vol. 16, no. 5, pp. 5451–5463, 2019.

[12] T. Fang, M. Jaggi and K. Argyraki, "Generating steganographic text with LSTMS," in *ACL 2017—55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, BC, Italy, pp. 100–106, 2017.

[13] Z. L. Yang, X. Q. Guo, Z. M. Chen and Y. F. Huang, "RNN-stega: Linguistic steganography based on recurrent neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1280–1295, 2018.

[14] H. Kang, H. Wu and X. Zhang, "Generative text steganography based on LSTM network and attention mechanism with keywords," *Electronic Imaging*, vol. 2020, no. 4, pp. 1–8, 2020.

[15] Z. Yang, S. Zhang, Y. Hu, Z. Hu and Y. Huang, "VAE-Stega: Linguistic steganography based on variational auto-encoder," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 880–895, 2021.

[16] L. Y. Xiang, S. H. Yang, Y. H. Liu, Q. Li and C. Z. Zhu, "Novel linguistic steganography based on character-level text generation," *Mathematics*, vol. 8, no. 9, pp. 1–18, 2020.

[17] I. J. Goodfellow, J. P. Abadie, M. Mirza and B. Xu, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 2, pp. 2672–2680, 2014.

[18] H. C. Shi, J. Dong, W. Wang, Y. L. Qian and X. Y. Zhang, "SSGAN: Secure steganography based on generative adversarial networks," in *Proc. Pacific Rim Conf. on Multimedia*, Harbin, China, pp. 534–544, 2017.

[19] D. H. Hu, L. Wang, W. J. Jiang, S. L. Zheng and B. Li, "A novel image steganography method via deep convolutional generative adversarial networks," *IEEE Access*, vol. 6, pp. 38303–38314, 2018.

[20] J. Liu, Y. Ke, Z. Zhang and Y. Lei, "Recent advances of image steganography with generative adversarial networks," *IEEE Access*, vol. 8, pp. 60575–60597, 2020.

[21] L. Yu, W. Zhang, J. Wang and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Proc. Thirty-first AAAI Conf. on Artificial Intelligence*, San Francisco, California, USA, pp. 2852–2858, 2017.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge: MIT Press, 1998.

[23] W. K. Hastings, "Monte carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[24] Z. Yang, N. Wei, Q. Liu, Y. Huang and Y. Zhang, "GAN-TStega: Text steganography based on generative adversarial networks," in *Proc. IWDW*, Chengdu, China, pp. 18–31, 2019.

[25] Y. Z. Hong, W. Feng and T. Peng, "t-distributed stochastic neighbor embedding (t-SNE) method with the least information loss for macromolecular simulations," *Journal of Chemical Theory and Computation*, vol. 14, no. 11, pp. 5499–5510, 2018.

[26] Z. Yang, W. Nan, J. Sheng, Y. Huang, Y. J. Zhang *et al.,* "TS-CNN: Text steganalysis from semantic space based on convolutional neural network," arXiv:1810.08136v1 [cs.CR], 2018.

[27] Z. Yang, K. Wang, J. Li, Y. Huang and Y. J. Zhang, "TS-RNN: Text steganalysis based on recurrent neural networks," *IEEE Signal Processing Letters*, vol. 26, no. 12, pp. 1743–1747, 2009.

[28] Z. Yang, Y. Huang and Y. J. Zhang, "A fast and efficient text steganalysis method," *IEEE Signal Processing Letters*, vol. 26, no. 4, pp. 627–631, 2019.

[29] J. Wen, X. Zhou, P. Zhong and Y. Xue, "Convolutional neural network based text steganalysis," *IEEE Signal Processing Letters*, vol. 26, no. 3, pp. 460–464, 2019.