Semantic-Preserving Linguistic Steganography by Pivot Translation and Semantic-Aware Bins Coding

Tianyu Yang, Hanzhou Wu, Member, IEEE, Biao Yi, Guorui Feng, and Xinpeng Zhang

Abstract—Linguistic steganography (LS) aims to embed secret information into a highly encoded text for covert communication. It can be roughly divided into two main categories, i.e., modification based LS (MLS) and generation based LS (GLS). MLS embeds secret data by slightly modifying a given text without impairing the meaning of the text, whereas GLS uses a well trained language model to directly generate a text carrying secret data. A common disadvantage for MLS methods is that the embedding payload is very small, whose return is well preserving the semantic quality of the text. In contrast, GLS enables the data hider to embed a large payload, which has to pay the high price of uncontrollable semantics. In this article, we propose a novel LS method to modify a given text by pivoting it between two different languages and embed secret data using a semantic-aware information-encoding strategy. Our purpose is to alter the expression of the given text, enabling a large payload to be embedded while keeping the semantic information unchanged. Experiments have shown that the proposed work not only achieves a large embedding payload, but also shows superior performance in maintaining the semantic consistency and resisting linguistic steganalysis.

Index Terms—Bins coding, data hiding, information hiding, linguistic steganography, pivot translation, semantic consistency.

I. INTRODUCTION

STEGANOGRAPHY [1] is referred to as the art of embedding secret information in a cover such as image without significantly distorting the cover. The resulting steganographic carrier containing hidden information does not introduce noticeable artifacts and will be sent to a receiver for information extraction via an insecure channel such as the Internet. The purpose of steganography is to hide the existence of the present secret communication so that secret information can be reliably conveyed to the receiver without arousing suspicion from the monitor. Steganography is not to replace cryptography but rather to enhance security using its concealment, promoting it to become quite important in modern information security.

The carrier used for steganography can be any media signal such as image [2] and video [3]. Among these media signals, text is very suitable for steganographic activity because text is the most commonly used information-carrier in our daily life. Especially, the rapid development of mobile social networks enables people to easily realize text steganography, which is also called linguistic steganography (LS) or natural language steganography (NLS). On the one hand, it is very convenient for the steganographer to integrate into social network through a mobile terminal. On the other hand, the seemingly-normal steganographic texts can be easily overshadowed by the huge number of ordinary texts. In other words, LS can be easily concealed by the huge number of normal activities over social networks. For the receiver, he can keep silent, collect steganographic texts and extract information from these texts without taking any suspicious interaction with any other user, which indicates that LS over social networks even conceals the real receiver. However, unlike other media signals that have a large redundant space, texts are highly encoded and therefore have small redundancy. This indicates that hiding information into texts is more challenging than other carriers.

A. Prior Arts

In recent years, with the rapid development of deep learning and natural language processing, increasing methods are proposed to realize LS, which can be roughly divided into two major categories, i.e., modification based LS (MLS) and generation based LS (GLS). MLS embeds secret information into a text that should be given in advance such as synonym substitution [4], [5], typo confusion [6] and syntactic transformation [7]. In contrast, GLS skips the given text and directly generates a text carrying secret information for steganography [8], [9], [10], [11], [12], [13].

To be specific, mainstream GLS methods use a good language model to generate steganographic texts, which can be generalized by two important steps. First, a language model is well trained on a large-scale corpus so that the trained language model can automatically generate texts with high semantic quality. Then, an information encoding method is used during text generation so that a sequence of words (carrying secret information) can be generated to form the steganographic text using the trained language model. A key problem is how to encode the secret information, i.e., how to build the mapping relationship between words and secret bits. Many methods are
carried out around this problem. For example, Fang et al. [12] divide a vocabulary into multiple subsets and select the token with the highest probability in the subset corresponding to the secret binary stream as the output. Yang et al. [10] propose two novel techniques called fix-length coding (FLC) and variable-length coding (VLC) based on the probability distribution of candidate words to map secret bits to words. Compared with MLS that should take into account the embedding impact on a given text, GLS does not require a given text, thereby providing a significantly larger payload.

Much attention has been paid to narrowing the difference in statistical distribution between steganographic texts and natural texts recently [13], [14]. However, for these methods, the term imperceptibility refers to language fluency and statistical similarity, rather than semantic consistency. Yang et al. [15], [16] propose novel strategies to make the semantic expression conform to the context, yet expressing the accurate semantics remains impossible, making it very difficult to adapt to realistic scenarios. Therefore, even though GLS has the larger payload, semantic consistency is still a very challenging problem.

On the other hand, MLS can be further divided into different categories. For example, word or phrase level MLS methods [4], [5] alter part of the cover text to embed secret information. These methods often first build a synonym dictionary to find the candidates, and then encode each candidate with various techniques such as binary tree [17] and Huffman coding [18]. However, word or phrase level MLS arts may cause ambiguity because of partial replacement, and they suffer the problem of small embedding payload. Sentence level MLS tends to convert a sentence into another form maintaining the same meaning, e.g., word ordering [19] and syntactic analysis [20]. Sentence level MLS modifies the entire sentence rather than part. Thus, it can maintain the semantic coherence of the entire sentence, which should pay the price of smaller embedding payload and can be detected by steganalysis tools [21], [22].

B. Our Contributions

We propose a sentence level MLS method based on a GLS-like information encoding method in this paper, which not only controls semantic consistency well, but also provides a large steganographic payload. The proposed method involves two key modules, i.e., pivot translation and SaBins encoding. Pivot translation aims at pivoting a natural text between two languages, i.e., English and German, to alter the entire expression of the natural text. The phrase “round-trip translation” is another expression of pivot translation. Both of them are used in previous paraphrase research. For consistency, we use “pivot translation” in this paper. Through pivot translation, we can solve the semantic inconsistency problem and the small embedding payload problem in the subsequent process. Besides, we propose a method called Semantic-Aware Bins (SaBins) to realize information encoding so that additional information can be embedded without impairing the semantics. Pivot translation and SaBins can be implemented independently. To this end, we first train a sequence-to-sequence-to-sequence (Seq2Seq2Seq) model to achieve the transformation of English-to-German-to-English (En2Ge2En), and then embed secret information in the decoding stage of the second sequence-to-sequence (Seq2Seq) model, i.e., German-to-English (Ge2En) model. We regard the German text as the unchanged semantic information of the natural text, and only change the English expression during pivoting. As a result, the steganographic text in English is different from the original natural text but has the same meaning. Due to semantic consistency and the fact that the receiver does not need the original text, the proposed method is quite suitable for application scenarios.

In summary, the main contributions of this paper include:

- By changing the expression of a given text through pivot translation and SaBins encoding, the proposed method achieves a very large payload and well maintains semantic consistency, which significantly outperforms mainstream MLS and GLS methods.
- Different from mainstream GLS methods that use a fixed-style corpus to train a language model and therefore have limited usage scenarios, the proposed method uses a self-supervised training strategy for paraphrase generation. It enables the data hider to use natural texts with any style for LS, which is more suitable for applications. Moreover, unlike many GLS methods requiring that the well-trained language model should be shared between the data hider and the data receiver in advance, the data receiver is able to extract secret data without the aid of language model in the proposed method, which is more helpful for use.
- Extensive experiments have indicated that, the proposed method not only achieves a larger payload compared with related works, but also shows superior performance in preserving the semantics and resisting steganalysis.
- The proposed LS method corresponds to a general framework. Novel advanced pivot translation strategies or Bins-like information encoding techniques can be applied to further improve the steganographic performance.

The remaining structure of this paper is organized as follows. We first present preliminaries in Section II. Then, we introduce the proposed work in detail in Section III, followed by convincing experimental results and analysis in Section IV. Finally, we conclude this work in Section V.

II. PRELIMINARY CONCEPTS

In this section, we briefly introduce related concepts so that we can better introduce the proposed method in Section III.

A. Sequence-to-Sequence (Seq2Seq) Model

As introduced in [23], a Seq2Seq model maps a sequence to another sequence where the length of the input sequence and the length of the output sequence may be different from each other. A Seq2Seq model generally consists of an encoder and a decoder. The encoder encodes the input sequence \( \mathbf{x} = \{x_1, x_2, ..., x_L\} \) (whose length is \( L > 0 \)) into a hidden vector. The decoder then calculates the output vector \( \mathbf{y}_t = \{y_{t,1}, y_{t,2}, ..., y_{t,N}\} \) at time \( t \) based on the hidden vector and the previous \( t - 1 \) output vectors \( \mathbf{y}_{1:(t-1)} \), where \( N \) equals the number of all possible values of each element to be generated, e.g., \( N \) is the number of all possible tokens (i.e., the size of the vocabulary) for generating a word sequence. By applying softmax to \( y_{t,i} \), we
can generate the present element by, for example, choosing the element corresponding to the highest prediction probability as the output. By collecting all generated elements, we are able to construct the final output sequence. A Seq2Seq model can be realized by using the long short-term memory [24] architecture or other cells. We refer the reader to [23] for more details.

B. Natural Language Translation (NLT)

Translation is a most basic task in natural language processing (NLP). The purpose of translation is to convert a text in one language to another with the same meaning. Traditional Statistical Machine Translation (SMT) systems [25], [26] are based on statistical models derived from a good corpus, well-designed linguistic rules, or a combination of them. With the rapid development of deep learning, most machine translation systems in use today are Neural Machine Translation (NMT) systems [27], [28]. In other words, translation can be treated as a Seq2Seq problem and realized by using a Seq2Seq model.

C. Byte Pair Encoding (BPE) and SubSeq

For text generation tasks, it is impossible to fill the vocabulary with all words because there are countless English words in the world and training a good text generation model will be extremely time consuming if the vocabulary is too large. However, if we use a small vocabulary, there will be many words that are not in the vocabulary and cannot be represented. Meanwhile, though twenty-six letters are enough to represent all English words, character-level tokenizer will result in too small granularity and make deep model training too difficult.

Byte pair encoding (BPE) and subword [29] are exploited to solve the above out-of-vocabulary (OOV) problem. It divides data into consecutive byte pairs and merges byte pairs according to the frequency in the training data, and finally forms a subword vocabulary. Subword is corresponding to a word segmentation method between character level and word level. A subword vocabulary consists of a large number of subwords and complete words. In recent research, BPE and subword have become one of the core technologies in NLP. In this paper, unless otherwise specified, we use the subword strategy based on BPE [29] for the proposed method.

D. Transformer

Transformer [30] has been widely used in the field of NLP in recent years due to the superiority in feature extraction and expression. Compared to previous Seq2Seq models [24], [31], [32], Transformer lies in parallel computing, which greatly reduces the time-consuming problem of Seq2Seq model training. The introduction of the position vector also prevents the loss of the relative position information between tokens.

Furthermore, thanks to attention mechanism [33], Transformer can automatically learn the correlation between words in the entire text. On the basis of attention mechanism, multi-head attention mechanism is introduced to help the model focus on several aspects of information and synthesize at the last step, which helps the network capture richer features.

E. Paraphrase Generation

Paraphrase is a method to explain the meaning of a given text in another expression using the same language. We can regard the semantic information of the paraphrase the same as the original text. Paraphrase generation is also a method of data argument in NLP, and has been used for performance improvement in several applications such as information extraction [34] and question answering [35].

Paraphrase generation is also related to linguistic steganography. Generating semantic-consistent text is a jointly goal of paraphrase and MLS. As a generation task, paraphrase can be the bridge between MLS and GLS. Considering using a specific dataset will lead to limited usage scenarios such as asking questions [36] and picture descriptions [37], we exploit a self-supervised method of generating paraphrases instead in this paper, so that any text can be used as a cover in our proposed method. We will show the details in Section III.

F. Bins Coding

Fang et al. [12] propose a novel GLS method that randomly partitions the vocabulary with a fixed size into $2^b$ bins evenly in advance, where $b$ is a pre-determined integer. Each token in the vocabulary belongs to exactly one bin. At each step, they select one bin based on the secret string. Then, they select the token with the highest conditional probability from the chosen bin. As a result, it embeds $b$ bits at each generation step. Notice that we ignore the common-token variant introduced in [12] throughout this paper unless mentioned.

The advantage of this novel method is that the data receiver neither needs to obtain the conditional probability, nor does he need the language model. Although there are many improved GLS methods afterwards such as [8], [9], [10], [11], they are subject to the constraint that the receiver has to calculate the conditional probability distribution for each token with the original model.

G. Synonym Substitution

The semantics of synonyms are similar to each other so that synonym substitution is a common method in MLS. It replaces the words in the cover text with synonymous words based on a dictionary and an information encoding strategy. For example, the two words “see” and “look” are corresponding to “0” and “1,” respectively. The word “see” can be used to replace the word “look” in a cover text in order to hide the bit “0”.

To prevent ambiguity, when using synonym substitution, a large natural corpus is used as supplement such as Google n-gram data [38] to evaluate the frequency of n-grams before and after the modified position in the large natural corpus. If the frequency exceeds a threshold [39] or is close to the frequency of original text [19], the replacement is considered reasonable.
Mainstream GLS methods require the data hider to share secret bits by generating a steganographic text according to the secret key and side information. Compared with previous works, the advantages of the proposed framework include:

- **Mainstream MLS methods embed secret bits using handcrafted substitution rules, which will result in a very small payload.** However, the proposed framework embeds data by changing the entire expression of a text in a generation way, which can provide a significantly larger payload.
- **Mainstream GLS methods embed secret bits by generating a steganographic text directly, which makes the semantics of the text uncontrollable.** However, the proposed framework pivots a text between two languages, which maintains the semantics very well and is more practical.
- **Mainstream GLS methods use a fixed-style corpus to train a language model.** It makes the style of the steganographic texts homogeneous. However, the proposed framework enables us to use a self-supervised strategy to train the language model such that texts with any style can be used for LS, which is more suitable for realistic scenarios.
- **Mainstream GLS methods require the data hider to share the trained language model with the data receiver so that secret data can be extracted at the receiver side.** However, in the proposed framework, the data receiver can extract secret data without accessing the trained language model, which significantly reduces the side information.

Based on the above analysis, we are now ready to describe the proposed method in detail in the following subsections.

### III. PROPOSED METHOD

#### A. General Framework

In order to provide a large embedding payload and keep the semantics unchanged, we propose to embed secret information into a given text by using pivot translation and SaBins encoding in this paper. As shown in Fig. 1, the proposed framework includes three phases, i.e., language encoding, language decoding and data extraction. The data hider will perform language encoding and language decoding, and the data receiver will perform data extraction. The goal of language encoding is to encode a given cover text in English into a new text in German (defined as pivot text). The two texts have different expressions but have the same meaning. During language decoding, the pivot text will be decoded into a steganographic text in English, whose semantic information is quite close to the original cover text and the pivot text. The steganographic text will be sent to the data receiver, who will try to reconstruct secret information from the steganographic text according to the secret key and side information. Compared with previous works, the advantages of the proposed framework include:

- Mainstream MLS methods embed secret bits using handcrafted substitution rules, which will result in a very small payload. However, the proposed framework embeds data by changing the entire expression of a text in a generation way, which can provide a significantly larger payload.
- Mainstream GLS methods embed secret bits by generating a steganographic text directly, which makes the semantics of the text uncontrollable. However, the proposed framework pivots a text between two languages, which maintains the semantics very well and is more practical.
- Mainstream GLS methods use a fixed-style corpus to train a language model. It makes the style of the steganographic texts homogeneous. However, the proposed framework enables us to use a self-supervised strategy to train the language model such that texts with any style can be used for LS, which is more suitable for realistic scenarios.
- Mainstream GLS methods require the data hider to share the trained language model with the data receiver so that secret data can be extracted at the receiver side. However, in the proposed framework, the data receiver can extract secret data without accessing the trained language model, which significantly reduces the side information.

Based on the above analysis, we are now ready to describe the proposed method in detail in the following subsections.

#### B. English-to-German-to-English (En2Ge2En) Model

We propose to use paraphrase generation to achieve LS. As mentioned previously, in order to satisfy the need to embed secret information in any cover text, we use a self-supervised paraphrase generation method instead of training on a specific paraphrase dataset. Inspired by [40], we generate paraphrase by pivoting between two different languages with NMT and select two proximate languages (i.e., English and German) as the pivot languages. We define it as an En2Ge2En model.

There are three reasons for using German as pivot language. First, pivot translation is a method to generate another English text through an intermediate language. In order to implement semantic-preserving LS, the gap between the generated text and the original text should be narrowed. One of the critical findings in pivot translation [40] is that “related languages are better,” meaning that using the related language, i.e., German, as the pivot language will introduce less semantic discrepancy. Second, using the language not from the same group will introduce more diversity. Diversity and semantic consistency constitute a pair of trade-off in language translation. However, in terms of practicality, we should not consider the diversity issue in pivot translation part in this steganography research. It is easier to introduce diversity than preserve the semantic meaning in the data embedding stage. If the data hider wants to introduce more diversity, he only needs to enlarge the payload of secret information. However, the payload will be the price when introducing more semantic consistency. Thus, semantic consistency should be emphasized as much as possible in the translation stage instead of the secret embedding stage, which also means that it is better to choose the language in the same group for pivot translation. Third, we adopt the popular WMT 2016 En2Ge dataset in the experimental section. This dataset is more commonly used in the translation task. We believe the experiments conducted based on such kind of dataset are more convincing than using some other pairs of languages.

As has shown in Fig. 1, the En2Ge2En model enables us to translate the cover text in English to another text in German, whose semantic information can be regarded as the same as the cover text. The text in German facilitates us to produce a new text in English, whose semantic information will be quite close to both the cover text and the text in German. Notice that the resulting text in English generated with the En2Ge2En model will not carry secret information if we do not embed secret information during the language decoding phase shown in Fig. 1.
It can be said that no matter we embed secret information or not, the final text (in English) generated with the En2Ge2En model is expected to have the same meaning to the cover text, and the pivot text in German shown in Fig. 1 can be regarded as the inner semantic meaning of the cover text in English.

It is necessary to specify the architecture of the model and its training strategy, which is the core research topic of NLP. Since it is not the main contribution of this paper and there are many advanced works in the literature, we use a most popular architecture called Transformer [30] as the implementation of the En2Ge2En model. The architecture details can be found in [30]. We are to describe how to train the En2Ge2En model in the following. It is pointed that training the En2Ge2En model is independent of embedding secret information.

The En2Ge2En model consists of two parts that are English-to-German (En2Ge) and Ge2En. As the basic translation task, we train the En2Ge model with a large corpus that contains a number of text pairs in English and German. However, when to train the Ge2En model, we create a new dataset. It is due to the reason that the pivot texts (in German) are no longer the original ones in the large corpus when to embed secret data, leading to the distribution difference between texts. Therefore, we use the trained En2Ge model to translate all English texts in the large corpus to generate the corresponding German texts. We then replace the original German texts in the large corpus with the generated German texts and keep the English texts unchanged to form a new dataset. Fig. 2 explains how to use two datasets to train the En2Ge and the Ge2En model. Except for the difference in dataset and input/output languages, we train the Ge2En model in the same fashion as the En2Ge model. We deem this method as a framework and any language pairs and models can be used to replace the English-German pair and Transformer structure. However, in actual use, it would be difficult to embed secret information in cover text which uses rare languages, because the text pairs of those less-used languages are more difficult to collect, thereby affecting the quality of model training. Thus, popular and approximate language pairs are recommended to use in the proposed framework.

C. Data Embedding

After training the entire En2Ge2En model, we are to use the trained model to embed secret data into the cover text. Unlike traditional MLS methods that modify a few words in the cover text, the proposed work alters the entire expression of the cover text in a generation way, whose payload is significantly larger.

As mentioned previously, the En2Ge2En model corresponds to a Seq2Seq2Seq model that enables us to output a new text by a word-wise (or called token-wise) fashion. In other words, the En2Ge2En model allows us to orderly generate a sequence of tokens (or called tokens) and construct a new text by orderly collecting all the words. Without the loss of generalization, let $y = \{y_1, y_2, \ldots, y_n\}$ represent the newly generated text, where $n > 0$ is the length of the generated text and $y_i$ means the $i$-th word. For consistency, we will also regard $y_1$ as a token. Let $V = \{v_1, v_2, \ldots, v_m\}$ represent the vocabulary including all tokens, where $m > 0$ is the size of the vocabulary. We can write $y_i \in V$ for all $i \in [1, n]$. The determination of $y_i$ can be briefly summarized as follows. According to the input, the Ge2En model (inside the En2Ge2En model) first outputs the prediction probability for each token in $V$. Formally, for each $y_i$, $i \in [1, n]$, the prediction probability of $v_j \in V$, $j \in [1, m]$ obtained by the Ge2En model is expressed as $p_{i,j} \in [0,1]$ and

$$\sum_{j=1}^{m} p_{i,j} = 1, \forall i \in [1, n].$$

A higher prediction probability implies that the corresponding token is more suitable to be selected as the present output. Generally, for each $y_i$, the token with the highest prediction probability can be selected as the output, i.e.,

$$y_i = \arg \max_{v_j \in V} p_{i,j}. \quad (2)$$

Accordingly, a sequence of tokens $y = \{y_1, y_2, \ldots, y_n\}$ can be generated regardless of the internal processing mechanism of the Ge2En model. Here, we have ignored the secret data, secret key, side information and information hiding strategy shown in Fig. 1. Moreover, if there are multiple $v_j$ satisfying the objective in (2), $y_i$ can be set to any one among them.

Now, in order to embed secret data, we should modify (2) in such a way that the present output (if it is used for data embedding) not only matches the secret data, but also results in a "good" text, i.e., the text $y$ containing secret data should be seemingly normal and semantically close to the cover text $c = \{c_1, c_2, \ldots, c_{n'}\}$, where $n' > 0$ is the length of the cover text and $c_i$ means the $i$-th token. Mathematically, given $V = \{v_1, v_2, \ldots, v_m\}$ and the corresponding prediction probabilities $p_i = \{p_{i,1}, p_{i,2}, \ldots, p_{i,m}\}$ satisfying (1), if $y_i$ is required to carry a secret stream $b_i \in \{0,1\}^l$, where $l_i \geq 0$ is the length of the secret stream, the data embedding operation for $y_i$ in this case can be written as
Finally, by collecting all the tokens that can be constructed in advance, the and 1 ≈. From which will be analyzed later, for the from Pseudocode for the Data Embedding Procedure.

Algorithm 1: Pseudocode for the Data Embedding Procedure.

Input: Cover text c, trained En2Ge2En model $\mathcal{M}$, secret data $b$, vocabulary $V$, secret key $k$, mapping function $f$.

Output: Steganographic text $y$.

1: Set a step value $s \geq 1$ according to $k$
2: Empty $y$ and then set $i = 1$
3: while need to generate the next token do
4: if $(i - 1) \mod s \neq 0$ or $b$ has been embedded then
5: Call (2) to determine $y_i$
6: else
7: Determine $b_i$ according to $b$ and all the previously embedded binary streams (if any)
8: Call (3) to determine $y_i$
9: end if
10: Append $y_i$ to $y$ and then set $i = i + 1$
11: end while
12: return $y$

the steganographic text by carefully manipulating the word prediction procedure. For example, the secret bits “010” are embedded into the third token by choosing the token “nice” as the output, which means that the data receiver can extract “010” from the steganographic token “nice” with necessary auxiliary data. The way to determine “nice” is briefly described in Fig. 4. First, as mentioned above, each token in $V$ is mapped to a binary stream by applying $f$. Then, given the input, the Ge2En model enables us to generate a prediction probability for each token in $V$. Finally, by collecting all the tokens that match the secret data, the token that has the highest probability among all the collected tokens is computed as the output, e.g., in Fig. 4, “nice” is chosen since it has the highest prediction probability among all the tokens matching “010”. In this way, the data embedding process for a single position is finished. By processing the remaining token positions, the entire secret data can be embedded. Despite the time complexity of constructing $f$ which will be analyzed later, for the $i$-th token position to be embedded, the time complexity to identify the required token is $O(|V|)$. However, since $f$ can be constructed in advance, the time complexity to identify the required token can be reduced to $O(|\mathcal{V}_b|)$, where $\mathcal{V}_b$ represents the set of tokens matching $b$, and can be determined in advance. Ideally, $|\mathcal{V}_b|/|V| \approx 1/2^n$. From the viewpoint of implementation, the time complexity can be further reduced by parallel computing since this search problem can be divided into multiple subproblems that do not affect each other. Therefore, the time complexity is low.

D. Data Extraction

It is easy for the data receiver to extract $b$ from $y$. According to the secret key, the data receiver first finds the step $s$. Then, with $f$, the data receiver determines out all the binary streams carried by the corresponding tokens in $y$. By concatenating all the binary streams, the entire secret data can be extracted. It is pointed that there is no need for the data receiver to hold the trained En2Ge2En model and the original cover text, which significantly reduces the side information shared between the data hider and the data receiver compared with mainstream GLS methods that require the data hider and the data receiver to share a trained language model in advance. Moreover, the proposed work enables the data hider to train language models that generate texts of various styles without sharing the models with anyone else, which prevents the trained language models.
Algorithm 2: Pseudocode for the SaBins Coding Method.

Input: \( V = \{v_1, v_2, \ldots, v_m\}, D_{1,0}, l, \) secret key \( k \).

Output: \( V_0, V_1, \ldots, V_{2l} \) (i.e., \( f \)).
1: Determine \( h = \{h(v_1), h(v_2), \ldots, h(v_m)\} \) based on \( D_{1,0} \)
2: Determine \( h' = \{h(v_{p_1}), h(v_{p_2}), \ldots, h(v_{p_m})\} \) by sorting \( h \) where we have \( h(v_{p_1}) \geq h(v_{p_2}) \geq \ldots \geq h(v_{p_m}) \geq 0 \) (the order can be controlled by \( k \) if these are multiple tokens that have the same frequency)
3: Initialize \( V_0 = V_1 = \ldots = V_{2l} = 0 \)
4: Mark all tokens in \( V \) as unprocessed
5: Set \( V_{2l} = \langle eos \rangle \) and then Mark \( \langle eos \rangle \) as processed
6: for \( i = 1, 2, \ldots, m \) do
7: Determine \( C_{p_i} \subset C \)
8: Collect all unprocessed tokens in \( C_{p_i} \) to build a set \( C_{p_i}' \), whose size is expressed as \( |C_{p_i}'| \)
9: while \( |C_{p_i}'| > 0 \) do
10: Determine \( n_s = \min(\{C_{p_i}'\}, 2^l) \)
11: Randomly select \( n_s \) tokens from \( C_{p_i}' \) by \( k \)
12: Randomly select \( n_s \) subsets from \( \{V_0, V_1, \ldots, V_{2l-1}\} \) by \( k \)
13: Randomly assign the \( n_s \) tokens to the \( n_s \) subsets so that no two tokens are in the same subset by \( k \)
14: Mark the \( n_s \) tokens as processed
15: Remove the \( n_s \) tokens from \( C_{p_i}' \)
16: end while
17: end for
18: return \( V_0, V_1, \ldots, V_{2l} \)

as privacy data from leaked. One thing to note is that the data hider and the data receiver should share \( f \) in advance, which can be constructed by an offline fashion and therefore will not require much computational cost. In addition, since \( f \) is only shared between the data hider and the data receiver, it will be very difficult for an attacker to extract secret data from \( y \).

E. Semantic-Aware Bins (SaBins) Coding

We propose an efficient method called SaBins to determine
\[
\begin{align*}
    f : V & \mapsto \mathbb{U}_{i=0}^{\infty} \{0, 1\}^4, \quad (4)
\end{align*}
\]
which maps each element in \( V \) to a binary stream. To this end, we divide \( V \) to \( 2^l + 1 \) disjoint subsets \( V_0, V_1, \ldots, V_{2^l} \), where \( l > 0 \) is a pre-determined integer. Mathematically, we have
\[
\begin{align*}
    V & = \bigcup_{i=0}^{2^l} V_i \quad \text{and} \quad V_i \cap V_j = \emptyset, \quad \forall 0 \leq i \neq j \leq 2^l. \quad (5)
\end{align*}
\]
For each \( i \in [0, 2^l] \), the tokens in \( V_i \) are mapped to the same stream corresponding to index \( i \) with a length of \( l \). For example, all the tokens in \( V_0 \) are mapped to “0011” if we have \( l = 4 \), i.e., \( f(e) = “0011” \) if \( e \in V_0 \) and \( l = 4 \). However, all the tokens in \( V_2 \) are mapped to an empty stream, i.e., all the tokens in \( V_2 \) will not carry secret information. Therefore, given the steganographic text \( y = \{y_1, y_2, \ldots, y_n\} \), if all the tokens in \( y \) are orderly used to carry secret information, we can retrieve the entire secret data by concatenating \( \{f(y_1), f(y_2), \ldots, f(y_n)\} \). A key problem is how to determine the “subsets,” i.e., “bins”.

The proposed SaBins method assigns the most special token \( \langle eos \rangle \) to \( V_{2l} \) only, i.e., \( V_{2l} \) contains one and exactly one token \( \langle eos \rangle \), which is generally the last token of each text indicating the end of the text. For example, the sentence “how are you?” generated by the language model represents the actual sentence “how are you?”. “\( \langle eos \rangle \)” enables us to decide when to stop text generation. “\( \langle eos \rangle \)” does not carry secret data. Now, our task is to construct \( V_0, V_1, \ldots, V_{2l-1} \).

One may simply assign each of the remaining tokens in \( V \) to a subset randomly selected from \( \{V_0, V_1, \ldots, V_{2l-1}\} \), which does not consider the semantic and distribution characteristics of tokens and may lead to very poor text quality. E.g., for tokens whose semantics are close to each other, and suit the present text generation step best, if they have been assigned to the same subset and do not match the secret stream to be embedded, it forces the text generation model to select an inappropriate token as the present output, which may result in very poor quality of the text. It motivates Fang et al. [12] to explore a variant where a set of common tokens are assigned to all the subsets so that the semantic quality of the text is high. However, these common tokens will no longer carry any secret information when they are included in all subsets. These tokens are used much more frequent than other words. As a result, there is a sharp decline for the embedding payload. To achieve a better trade-off between text quality and embedding payload, in this paper, each of the remaining tokens in \( V \) will be assigned to a subset from \( \{V_0, V_1, \ldots, V_{2l-1}\} \) based on its distribution characteristics in the dataset \( D_1 \) in Fig. 2. We use \( D_{1,0} \) to denote the set including all the English texts in \( D_1 \).

Specifically, we first determine the frequency of each token \( v_i \in V \) appeared in \( D_{1,0} \) expressed as \( h(v_i) \geq 0 \). It is noted that \( V \) can be determined by collecting all the different tokens in \( D_{1,0} \). Then, all these tokens are sorted in the descending order according to the frequencies. Without the loss of generalization, let \( \{v_{p_1}, v_{p_2}, \ldots, v_{p_m}\} \) be the sorted sequence, where \( h(v_{p_1}) \geq h(v_{p_2}) \geq \ldots \geq h(v_{p_m}) \) and \( \{p_1, p_2, \ldots, p_m\} \) is a permutation of \( \{1, 2, \ldots, m\} \). For each token \( v_p \), except for \( \langle eos \rangle \), in the sorted sequence, we determine its substitution set \( C_{p_i} \), which means that the corresponding token \( v_p \) can be replaced by any token in \( C_{p_i} \) to keep the text semantically well.

Obviously, the token \( v_p \) itself surely belongs to \( C_{p_i} \), i.e., \( v_p \in C_{p_i} \). For \( C_{p_i} \), we will assign every unprocessed token \( e \in C_{p_i} \) to a randomly selected subset \( V_j \). By orderly processing \( \{C_{p_1}, C_{p_2}, \ldots, C_{p_m}\} \), we can construct \( \{V_0, V_1, \ldots, V_{2l-1}\} \). With \( \{V_0, V_1, \ldots, V_{2l-1}\} \) and \( V_{2l} \), we actually finish the construction of \( f \). Algorithm 2 shows the detailed pseudocode. We can find that the construction of the subsets considers the semantic and the distribution characteristics of tokens. That is why we define it as semantic-aware coding.

For better explanation, we provide a simplified but intuitive example for SaBins coding in Fig. 5. In Fig. 5(a), four tokens \( v_1, v_2, v_3, v_4 \) are collected and their substitution sets are sorted according to the inequality \( h(v_1) > h(v_2) > h(v_3) > h(v_4) \). Suppose that \( l = 1 \), each of the four tokens should be randomly
is randomly assigned to itself and $\times l\ | v_1$ in our simulation experiments. Therefore, the $(e)$ is actually equivalent to a codebook, which can be directly $C$ divides the length of $\times l$ is a small coefficient because max $\times m$ cannot be fully embedded, $\times l$ includes $= 1$ for token assignment, which $O$ (BiLingual Evaluation Understudy) $\approx v_4, v_1, v_2$.

Even though the construction of $1$ cannot be fully embedded, $\times l$ is used to carry the $f$, a synonym graph can be constructed according to the prediction probabilities. However, if the data hider wants to embed “1,” $v_1$ or $v_4$ will be used as the output according to the prediction probabilities.

**Remark 1:** We use the synonym relationship in WordNet\(^1\) to determine $C_{p_i}$ appeared in Line 7 in *Algorithm 2*. WordNet contains around $1.1 \times 10^5$ synonym sets. The tokens in each synonym set have the same meaning. A token may appear in multiple synonym sets. For a token $v_{p_i}$, we collect all tokens in synonym sets that contains $v_{p_i}$ to constitute the corresponding substitution set $C_{p_i}$. Even though the construction of $C_{p_i}$ can be optimized (as all tokens in $C_{p_i}$ are not semantically related to each other), it is not the main interest of this paper and our experiments show that the above construction has excellent performance. We leave the optimization task for future work. To determine $C_{p_i}$, a synonym graph can be constructed according to the synonym sets. In the graph, the nodes are tokens and the edges represent the synonym relationship between tokens. Therefore, $C_{p_i}$ can be easily determined by collecting $v_{p_i}$ itself and its adjacent nodes. After determining $C_{p_i}$, we collect all the unprocessed tokens in $C_{p_i}$ for token assignment, which requires a time complexity of $O(|C_{p_i}|)$. If all the possible $C_{p_i}$ have been previously determined, the overall time complexity for the token assignment is $O(\sum_{i=1}^{m} |C_{p_i}|) = O(km)$, where $k << m$ is a small coefficient because max$_i |C_{p_i}| << m$, e.g., max$_i |C_{p_i}| \approx 10$ in our simulation experiments. Therefore, the time complexity is satisfactory for realistic scenarios. In our experiments, we will further provide run-time analysis.

**Remark 2:** The side information shown in Fig. 1 includes $f$ and all the pre-determined parameters such as $s$ and $l$. All these side information can be controlled by using a secret key shared between the data hider and the data receiver. In other words, by using the secret key shared in advance, the data hider and the data receiver can generate the same side information used for data embedding and data extraction, which has been widely applied in secure communication. It is worth mentioning that $f$ is actually equivalent to a codebook, which can be directly shared between the data hider and the data receiver in advance. Alternatively, since the SaBins coding procedure is public to the data hider and receiver, by calling the SaBins coding procedure with the same parameters, the receiver can find $f$.

**Remark 3:** For the Line 8 in *Algorithm 1*, $b_i$ has a length of $l$. Its determination is based on $b$ and all the previously embedded streams. For example, if $b = "010111000"$ and the previously embedded stream is “010,” the present stream to be embedded should be “111”. We assume that $l$ divides the length of $b$, which is reasonable since we can always append “0” to $b$ to guarantee that $l$ divides the length of $b$. If $b$ cannot be fully embedded, multiple steganographic texts can be used.

### IV. Performance Evaluation and Analysis

#### A. Dataset and Model

In the experiments, we train the En2Ge model on the WMT 2016 English-German translation dataset consisting of around $4.5 \times 10^6$ sentence-pairs and a vocabulary containing around $3.3 \times 10^4$ tokens (or called words) \([41]\). The official dataset has been split to three disjoint subsets called training set, validation set, and testing set. We randomly choose some sentence-pairs from the training set and move them to the testing set so that the total number of natural English texts in the testing set is exactly equal to 10,000. After training the En2De model with the training set and the validation set, by inputting the original English texts in the two sets to the trained En2De model, we collect all the generated German texts. Thereafter, we train the De2En model according to the original English texts in the two sets and the generated German texts.

As mentioned in Section III-B, we train both the En2Ge model and the Ge2En model with the structure of Transformer, by exploiting the sequence modeling toolkit Fairseq\(^2\). The rest of the training details are consistent with \([30]\). To evaluate the translation performance, we decode the translated text with a beam size of 4 and select the best candidate as the result. The BLEU (BiLingual Evaluation Understudy) \([42]\) is then used to evaluate the machine-translated texts. Experimental results

---

\(^1\)https://wordnet.princeton.edu/

\(^2\)https://www.statmt.org/wmt16/

\(^3\)https://github.com/pytorch/fairseq
show that the BLEU scores for En2Ge and Ge2En are 27.83 and 51.10. It implies that the two well-trained models provide satisfactory translation performance. It is pointed that training the two models is independent of data embedding.

Once again, pivot translation only serves as a framework and various datasets and model structures can be implemented on this framework. The quality of dataset and model will surely influence the performance of pivot translation, and further influence the quality of steganographic texts. We would like to explore such influence in future works.

B. Evaluation Metrics

1) Bits per Word: We use bits per word (BPW) to measure the amount of embedded information, which is determined as the ratio between the number of embedded bits and the number of tokens excluding "<eos>" in the original cover text. We do not count the number of tokens in the steganographic text since even the same cover text will result in different steganographic texts with a different number of tokens by applying different parameters. BPW can be controlled by two parameters \( s \) and \( l \). Generally, BPW will increase when \( s \) decreases or \( l \) increases. In experiments, \( s = \infty \) and \( l = 0 \) means to choose the token with the highest prediction probability as the present output at each generation step, i.e., there is no embedded information (i.e., BPW = 0), which serves as a baseline for fair comparison.

2) BLEU: BLEU [42] is a metric commonly used to evaluate the text quality in machine translation tasks. It counts the coincidence frequency of n-gram words between the machine-generated texts and the gold standard. We evaluate the quality of steganographic texts with BLEU score due to the use of translation models in our method. Obviously, a higher BLEU score means that the steganographic texts are more similar to the cover texts, i.e., the text quality is better. However, it should be pointed that our primary task is not translation.

3) BERTScore: BERTScore [43] is another metric to evaluate the similarity between texts. Because regarding a large number of partial overlaps between texts as semantic similarity is one-sided, BERTScore takes the internal meaning of the texts into consideration. It calculates the similarity according to word vectors between texts. Because the vectors of words with approximate meaning are also similar to each other, BERTScore is able to evaluate in the semantic space.

4) Perplexity (PPL): The above two metrics are designed to evaluate the similarity of the texts, while paying less attention to the fluency. For example, altering the order of words may raise little influence on BLEU and BERTScore, but may cause critical unnaturalness and is confusing. Following [44], we evaluate the fluency of steganographic texts by PPL, which is defined as the exponential average negative log-likelihood of a token sequence with a pre-trained language model unless otherwise specified. According to its definition, simply speaking, when tokens with higher probabilities are selected during text generation, the resulting PPL will be smaller, meaning that the generated text is more fluent.

5) Steganalysis Accuracy: Steganalysis evaluates the security of steganographic algorithms, which is to detect whether there is secret information from the observed data. In experiments, we assume that the detection part is available to labeled data. We analyze the anti-steganalysis ability of our method under such adverse condition. Steganalysis can be regarded as a classical binary classification problem, that is, the closer the accuracy rate is to 0.5, the harder for the detection part to distinguish between steganographic texts and natural ones.

Following [45], we finetune BERT-base-cased with the default settings in Hugging Face Transformers\(^4\) as the detection part unless otherwise specified. For each experiment, we randomly split 10,000 original natural texts and their steganographic versions to three disjoint subsets: training set (60%), validation set (10%), and testing set (30%). The discriminators are trained for 60 epochs, with a batch size of 32. The learning rate is \(10^{-6}\). Adam [46] is the optimizer. The accuracy is evaluated on the testing set with the model of the highest validation accuracy. It is noted that the detection accuracy is still much higher than 0.5 even BPW = 0, indicating that the difference between steganographic texts and cover texts not only comes from data embedding, but also partly from the model [47].

C. Performance Comparison With Baseline Method

We propose a novel method called SaBins for information encoding. Unlike Bins [12] that randomly partitions the vocabulary with a fixed size into a certain number of bins evenly in advance, SaBins takes into account the semantic information of tokens and distributes tokens with the similar meaning into different bins so that tokens more fitting into the context can be used for text generation while carrying secret information. In order to verify the superiority of SaBins, we compare SaBins with Bins in terms of different indicators. Table I shows the experimental results. In Table I, for Bins, for example, \( l = 2 \) means to randomly partition the vocabulary into \( 2^2 = 4 \) bins. One more thing to note is that both Bins and SaBins in Table I use pivot translation for fair comparison. In other words, the

\[\text{Table I: Performance Comparison Between Bins (Baseline) and SaBins (Proposed) Due to Different Parameters}\]

| Parameters | Method | BLEU | BERTScore | PPL | Accuracy |
|------------|--------|------|-----------|-----|----------|
| \( s = \infty, l = 0 \) | Bins | 47.60 | 0.9555 | 1.219 | 0.4922 ± 0.0163 |
| | SaBins | 21.83 | 0.9155 | 2.543 | 0.8381 ± 0.0076 |
| \( s = 3, l = 1 \) | Bins | 16.02 | 0.9034 | 3.315 | 0.6638 ± 0.0090 |
| | SaBins | 19.10 | 0.9083 | 3.096 | 0.8458 ± 0.0043 |
| \( s = 3, l = 2 \) | Bins | 12.35 | 0.8942 | 3.734 | 0.5910 ± 0.0137 |
| | SaBins | 14.87 | 0.8999 | 3.616 | 0.8834 ± 0.0050 |
| \( s = 1, l = 1 \) | Bins | 6.86 | 0.8782 | 6.563 | 0.9105 ± 0.0060 |
| | SaBins | 8.58 | 0.8840 | 6.019 | 0.8960 ± 0.0100 |
| \( s = 2, l = 2 \) | Bins | 6.70 | 0.8776 | 6.042 | 0.9176 ± 0.0063 |
| | SaBins | 8.59 | 0.8835 | 5.530 | 0.9092 ± 0.0038 |
| \( s = 3, l = 3 \) | Bins | 7.02 | 0.8768 | 5.724 | 0.9167 ± 0.0061 |
| | SaBins | 8.82 | 0.8799 | 5.710 | 0.9068 ± 0.0055 |
| \( s = 2, l = 3 \) | Bins | 2.84 | 0.8335 | 9.779 | 0.9522 ± 0.0037 |
| | SaBins | 3.90 | 0.8620 | 9.637 | 0.9485 ± 0.0038 |
| \( s = 1, l = 2 \) | Bins | 1.88 | 0.8491 | 15.665 | 0.9560 ± 0.0030 |
| | SaBins | 1.51 | 0.8440 | 16.327 | 0.9778 ± 0.0043 |
| \( s = 1, l = 3 \) | Bins | 0.38 | 0.8235 | 33.105 | 0.9977 ± 0.0047 |
| | SaBins | 0.43 | 0.8252 | 28.557 | 0.9786 ± 0.0036 |

\(4\)https://huggingface.co/transformers/
difference between Bins and SaBins in Table I is that their information encoding strategies are different from each other while the other settings are same as each other. In Table I, for BLEU, BERTScore and PPL, we collect the corresponding average values as the results, e.g., each steganographic text has a PPL and the average value of all PPLs can be determined as the result. For steganalysis, in each experiment, we use ten trained models to determine the detection accuracy. The mean and standard deviation are determined. It can be observed that SaBins outperforms Bins in all cases, which has verified the superiority of our work. It is inferred that different parameters may correspond to the same payload, e.g., the BPW equals 1 if $s = l$ (i.e., every $s$ tokens carry $l = s$ bits averagely). For a fixed $l$, when $s$ increases, the text quality will become better. The reason is that by using a larger $s$, secret information can be evenly distributed throughout the entire text, rather than gather in a certain local area, which is conducive to achieving better text quality. In contrast, for a fixed $s$, when $l$ increases, the text quality will become worse because more bits are carried by the text. Overall, by fine-tuning the parameters, a good trade-off between payload, text quality and security can be obtained.

D. Performance Comparison With Mainstream Methods

It is necessary to compare the proposed method with MLS since both have the same purpose in maintaining the semantic consistency. Most traditional MLS methods rely on substitute rules or hash mapping, which may lead to a problem that the embedding strategy is unavailable when there are no matching rules or mapping. Table II shows the performance comparison between [49] and the proposed method. In [49], Wilson et al. exploit both substitute rules with Paraphrase DataBase [50] and hash mapping. The embedding success rate cannot reach 100%. Alternatively speaking, the success rate also reflects the problem of small embedding payload in MLS methods. It is seen that the embedding success rate is close to 100% when the embedding payload is relatively small. However, the proposed method gets rid of such restriction. The steganographic texts can be generated based on any given cover texts. The method in [49] shows satisfactory results in BLEU and BERTScore because of the partial replacement based strategy. However, the result in steganalysis accuracy exposes its unnaturalness and turns out that our method is more ideal in terms of security.

It is also necessary to compare the proposed method with GLS because both embed secret data in a generative fashion. However, GLS generates the steganographic texts without a cover text so that it is difficult to analyze the quality of the steganographic texts. To solve this problem, we evaluate the quality of the steganographic text by comparing it with the corresponding zero-bit text, which is generated by the same model but without any hidden information. The zero-bit text is generated by always choosing the token with the highest prediction probability as the present output during text generation. Because RNN-Stega [10] and Bins [12] do not require any input text, they need to be implemented on a language model. For fair comparison, we fine-tune another GPT-2 [51] model with the generated English texts mentioned in Section IV-A as the language model to conduct experiments. We apply the parameters $s$ and $l$ to the GLS methods so that we can make comparison under the condition of same BPW. We use the fixed-length coding (FLC) in [10] for simulation. For Bins, the baseline strategy in Table I is used. 10,000 steganographic texts and the corresponding zero-bit texts are used for each steganalysis experiment. During text generation, we select the tokens with top-10000 appearance frequency in the training set mentioned in Section IV-A as the first tokens of the steganographic texts so that no two steganographic texts are the same as each other, which has been applied in existing methods. As shown in Table III, our method better preserves the semantic information in text generation compared to GLS methods. It turns out that the steganographic texts generated by our method are more approximate to natural texts, which is consistent with the requirement of higher security and verified by the steganalysis results.

To further verify the superiority, we apply different steganalysis methods to the LS methods, and compute the corresponding detection accuracy for each experiment for analysis. Tables IV and V provide the results, where the former compares [49] and the proposed method, while the latter compares GLS methods and the proposed method. It is inferred from the two Tables that different steganalysis methods result in different detection accuracies, which is in line with expectation because of different model structures. However, the regular remains the same despite the change of steganalysis technique, that is, the proposed method outperforms the related works significantly in terms of anti-steganalysis.

E. Qualitative Results

We provide some examples to evaluate steganographic texts generated by the proposed method. Table VI shows the results. It indicates that when BPW rises, not only the performance on various metrics will decrease, but also the generated text is less grammatical and semantic consistent. Such trade-off between payload and quality of steganographic texts is consistent with the result shown in Table I. Although some texts carrying more secret information seem more suspicious, those large BPW settings are still meaningful because the text quality is not the only metric in steganography. However, from the viewpoint of security, when the sender needs to escape from strict detection, a small BPW is recommended. Otherwise, the sender can feel free to choose a large BPW to pursue embedding efficiency. We visualize the statistical distribution of steganographic texts and natural texts by using t-SNE [48]. As shown in Fig. 6, more points of the two colors overlap as BPW decreases, implying

| Method  | Success Rate | BLEU | BERTScore | Accuracy |
|---------|--------------|------|-----------|----------|
| [49]    | 93.07%       | 85.32 | 0.3753    | 0.8383 ± 0.0095 |
| Proposed| 100%         | 25.92 | 0.9213    | 0.8111 ± 0.0047 |
that the steganographic texts are more approximate to the natural ones and thus have high security.

F. Other Comparisons and Complexity Analysis

Our simulation is based on the subword strategy, which has been mentioned in Section II-C. Dividing the words into subwords will reduce the number of synonym relationships in the vocabulary. As a result, in our simulation, the number of synonyms relationship decreases from around $2.6 \times 10^4$ to around $9.4 \times 10^3$ for the used dataset. However, the text quality shows comparable results with the condition of word-level. That is because only those words that appear less frequently will be divided into subwords. Due to the low occurrence rate, their influence on SaBins is negligible. However, when implementing the secret embedding strategy on a word-level Seq2Seq2Seq model, the security drops sharply due to a large number of unknown tokens as shown in Fig. 7. As shown in Table VII, it is also confusing, which is more likely to arouse the suspicion of the detector. Generally speaking, embedding in a subword-level is superior to that in a word-level, despite of the reduction of synonym relationships.

The authors in [12] propose a novel common-token strategy for improving the quality of the steganographic text. The price of this strategy is that the used common tokens no longer carry secret information, resulting in a smaller payload. We select the tokens with top-1000 appearance frequency in the training set mentioned in Section IV-A as common tokens and introduce the same parameters in Table III for fair comparison. As shown in Table VIII, although applying common-token improves the text quality sightly compared with “GPT-2+Bins,” it is still at a low level and more likely detected by the steganalyzer compared with the proposed method, which has demonstrated the superiority of the proposed method.

In addition, the computational complexity of the proposed method is mainly affected by model training and SaBins. Since both can be completed by an off-line fashion, the proposed method is very suitable for application scenarios based on the aforementioned experiments. Regardless of the computational complexity of off-line model training, we provide the run-time comparison in Table IX from other aspects. In our experiments, the CPU is Intel(R) Xeon(R) Gold 5118 CPU@2.30 GHz, and the GPU is NVIDIA TITRAN RTX 24GBx1. The entire secret hiding process can be divided to auxiliary data loading part and data embedding part. The former part can be further divided into model loading and mapping construction. Before detailed introduction, it is noted that auxiliary data can be only loaded once and then repeatedly used to embed secret information. Language model is not required in [49]. And using substitution and hash mapping to embed data is much faster than other methods. However, it consumes a lot of time to load the
substitution rules. Because the replacement rule needs to be read only once, it is also applicable when to be used for many times. GPT-2+FLC and GPT-2+Bins belong to GLS. Since the language models used are the same, the time for loading the language models should be the same theoretically, but there are always slight differences in practical use. It is not required for FLC to load the mapping relationship in advance, while Bins randomly assigns the tokens. Therefore, the time of building the mapping relationship is negligible. Since they share the same generation strategy and only use different sampling rules, the time consumption is similar for steganographic text generation.

The proposed method adopts the Seq2Seq2Seq structure. The more complex structure makes the method consume more time. The number of parameters in each Seq2Seq model is larger than that in the GPT-2 model. Thus, the proposed method takes more than twice as long as other GLS methods to load the model and generate the steganographic text. Meanwhile, SaBins needs
to load the synonym relationship and take it into mapping construction, which requires auxiliary time. Therefore, the proposed method does require more time. But it is valuable to trade a little time consumption for better performance, which has been proved in previous experiments, because security and payload are the core issues in steganography.

V. CONCLUSION

LS is not only promising but also challenging. Mainstream MLS methods suffer the problem of small embedding payload, while GLS methods are restricted by semantic inconsistency. In this paper, we propose a novel LS method based on pivot translation to maintain the semantic meaning in an automatic generation manner with a Seq2Seq2Seq model and provide a large payload. We also propose an efficient information encoding method to improve the text quality. Extensive experiments have shown that the proposed method outperforms the baseline method, mainstream MLS methods and GLS methods in terms of various indicators. It is believed that the proposed method has good potential in applications. In future, we will improve our method so that the security can be further enhanced.

ACKNOWLEDGMENTS

The authors would like to sincerely acknowledge the Editor and the Anonymous Reviewers for their valuable comments and suggestions for improving the article.

REFERENCES

[1] G. J. Simmons, “The prisoners’ problem and the subliminal channel,” in Advances in Cryptology. Berlin, Germany: Springer, 1984, pp. 51–67.
[2] H. Wu, Y. Shi, H. Wang, and L. Zhou, “Separable reversible data hiding for encrypted palette images with color partitioning and flipping verification,” IEEE Trans. Circuits Syst. Video Technol., vol. 27, no. 8, pp. 1620–1631, Aug. 2017.
[3] Y. Chen, H. Wang, H. Wu, Z. Wu, T. Li, and A. Malik, “Adaptive video data hiding through cost assignment and STCS,” IEEE Trans. Dependable Secure Comput., vol. 18, no. 3, pp. 1320–1335, May/Jun. 2021.
[4] K. Weinstein, “The secret’s in the word order: Text-to-text generation for linguistic steganography,” in Proc. Int. Conf. Comput. Linguistics, 2012, pp. 511–528.
[5] R. Bergmaier, “Natural language steganography and an AI-complete security primitive,” in Proc. 21st Chaos Commun. Congr., 2014, pp. 1–9.
[6] C.-Y. Chang and S. Clark, “A confusing approach,” in Proc. Int. Conf. Comput. Linguistics, 2012, pp. 1045–1048.
[7] T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2015, pp. 1412–1421.
[8] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.
[9] M. Osborne, “Motivating language model for efficient linguistic steganalysis,” 2021, arXiv:2107.12168.
[10] H. Wu, B. Yi, F. Ding, G. Feng, and X. Zang, “Exploiting language model for efficient linguistic steganalysis,” 2021, arXiv:2107.12168.
[11] Z. Yang, S. Zhang, Y. Hu, Z. Hu, and Y. Huang, “VAE-Stega: Linguistic steganography based on variational auto-encoder,” IEEE Trans. Inf. Forensics Secur., vol. 16, pp. 880–895, Sep. 2020.
[12] T. Fang, M. Jaggi, and K. Argyraki, “Generating steganographic text with LSTM,” 2017, arXiv:1705.10342.
[13] Z. Ziegler, Y. Deng, and A. Rush, “Neural linguistic steganography,” 2019, arXiv:1909.01496.
[14] F. Z. Dai and Z. Cai, “Towards near-imperceptible steganographic text,” in Proc. 57th Annu. Meeting Assoc. Comput. Linguistics, 2019, pp. 4303–4308.
[15] Z. Yang, B. Gong, Y. Li, J. Yang, Z. Hu, and Y. Huang, “Graph-Stega: Semantic controllable steganographic text generation guided by knowledge graph,” 2020, arXiv:2006.08339.
[16] Z. Yang, L. Xiang, S. Zhang, X. Sun, and Y. Huang, “Linguistic generative steganography with enhanced cognitive-imperceptibility,” IEEE Signal Process. Lett., vol. 28, pp. 409–413, Feb. 2021.
[17] Y.-L. Chiang, L.-P. Chang, W.-T. Hsieh, and W.-C. Chen, “Natural language watermarking using semantic substitution for Chinese text,” in Proc. Int. Workshop Digit. Watermarking, 2003, pp. 129–140.
[18] R. Bergmaier, “Natural language steganography and an AI-complete security primitive,” in Proc. 21st Chaos Commun. Congr., 2014, pp. 1–9.
[19] C.-Y. Chang and S. Clark, “Analyzing the secret’s word order: Text-to-text generation for linguistic steganography,” in Proc. Int. Conf. Comput. Linguistics, 2012, pp. 511–528.
[20] B. Murphy and C. Vogel, “The syntax of concealment: Reliable methods for plain text information hiding,” in Proc. SPIE Int. Soc. Opt. Eng. Secur. Steganogr. Watermarking Multimedia Contents IX, 2007, pp. 351–362.
[21] B. Yi, H. Wu, G. Feng, and X. Zang, “Exploiting language model for efficient linguistic steganalysis,” 2021, arXiv:2107.12168.
[22] M. Osborne, “Statistical Machine Translation.” Cambridge, U.K.: Cambridge Univ. Press, 2009.
[23] T. Fang, M. Jaggi, and K. Argyraki, “Generating steganographic text with LSTM,” 2017, arXiv:1705.10342.
[24] L. Dong, J. Mallinson, S. Reddy, and M. Lapata, “Learning to paraphrase with neural networks,” 2014, arXiv:1409.1259.
[25] M. Osborne, “Motivating language model for efficient linguistic steganalysis,” 2021, arXiv:2107.12168.
[26] B. Murphy and C. Vogel, “The syntax of concealment: Reliable methods for plain text information hiding,” in Proc. SPIE Int. Soc. Opt. Eng. Secur. Steganogr. Watermarking Multimedia Contents IX, 2007, pp. 351–362.
[27] B. Yi, H. Wu, G. Feng, and X. Zang, “Exploiting language model for efficient linguistic steganalysis,” 2021, arXiv:2107.12168.
[28] T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2015, pp. 1412–1421.
[29] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.
[30] M. Osborne, “Motivating language model for efficient linguistic steganalysis,” 2021, arXiv:2107.12168.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
[39] C.-Y. Chang and S. Clark, “Linguistic steganography using automatically generated paraphrases,” in Proc. Hum. Lang. Technol. Ann. Conf. North Amer. Chapter Assoc. Comput. Linguistics, 2010, pp. 591–599.

[40] C. Federmann, O. Elachqar, and C. Quirk, “Multilingual whispers: Generating paraphrases with translation,” in Proc. 5th Workshop Noisy User-Generated Text, 2019, pp. 17–26.

[41] Y. Wu et al., “Google’s neural machine translation system: Bridging the gap between human and machine translation,” 2016, arXiv:1609.08144.

[42] K. Papineni, S. Roukos, T. Ward, and W. Zhu, “BLEU: A method for automatic evaluation of machine translation,” in Proc. 40th Annua. Meeting Assoc. Comput. Linguistics, 2002, pp. 311–318.

[43] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTScore: Evaluating text generation with BERT,” 2019, arXiv:1904.09675.

[44] X. Zhou, W. Peng, B. Yang, J. Wen, Y. Xue, and P. Zhong, “Linguistic steganography based on adaptive probability distribution,” IEEE Trans. Dependable Secure Comput., vol. 19, no. 5, pp. 2982–2997, Sep./Oct. 2022.

[45] W. Peng, J. Zhang, Y. Xue, and Z. Yang, “Real-time text steganalysis based on multi-stage transfer learning,” IEEE Signal Process. Lett., vol. 28, pp. 1510–1514, Jul. 2021.

[46] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.

[47] S. Zhang, Z. Yang, J. Yang, and Y. Huang, “Provably secure generative linguistic steganography,” 2021, arXiv:2106.02011.

[48] A. C. Belkina, C. O. Ciccolella, R. Anno, R. Halpert, J. Spidlen, and J. E. Snyder-Cappione, “Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets,” Nature Commun., vol. 10, no. 1, pp. 1–12, 2019.

[49] A. Wilson and A. D. Ker, “Avoiding detection on Twitter: Embedding strategies for linguistic steganography,” in Proc. SPHE Electron. Imag. Media Watermarking Secur. Forensics, 2016, pp. 1–9.

[50] E. Pavlick, P. Rastogi, J. Gantitkevitch, B. V. Durme, and C. Callison-Burch, “PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification,” in Proc. 33rd Annua. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process., 2015, pp. 425–430.

[51] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” OpenAI Blog, vol. 1, no. 8, 2019, Art. no. 9.

[52] Z. Yang, K. Wang, J. Li, Y. Huang, and Y. Zhang, “TS-RNN: Text steganalysis based on recurrent neural networks,” IEEE Signal Process. Lett., vol. 26, no. 12, pp. 1743–1747, Dec. 2019.

[53] Z. Yang, Y. Huang, and Y. Zhang, “TS-CSW: Text steganalysis and hidden capacity estimation based on convolutional sliding windows,” Multimedia Tools Appl., vol. 79, no. 25, pp. 18293–18316, 2020.

Tianyu Yang received the BS degree from Shanghai University, China, in June 2022, where he is currently working toward the MS degree. His research interests include linguistic steganography and natural language processing.

Biao Yi received the BS degree from the University of South China, China, in June 2019, and the MS degree from Shanghai University, China, in June 2022. He is currently working toward the PhD degree with Nankai University, China. His research interests include natural language processing, information hiding, and artificial intelligence security.

Guorui Feng received the BS and MS degrees in computational mathematics from Jilin University, Changchun, China, in 1998 and 2001, respectively, and the PhD degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2005. From January 2006 to December 2006, he was an assistant professor with East China Normal University, Shanghai. In 2007, he was a research fellow with Nanyang Technological University, Singapore. He is currently with the School of Communication and Information Engineering, Shanghai University.

Xinpeng Zhang received the BS degree from Jilin University, China, in 1995, and the MS and PhD degrees from Shanghai University, in 2001 and 2004, respectively. Since 2004, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a full-time professor. He is also with the faculty of the School of Computer Science, Fudan University. He was with the State University of New York at Binghamton as a visiting scholar from 2010 to 2011, and also with Konstanz University as an experienced researcher, sponsored by the Alexander von Humboldt Foundation from 2011 to 2012. His research interests include multimedia security, image processing, and digital forensics. He has published more than 200 research papers. He was an associate editor of the IEEE Transactions on Information Forensics and Security from 2014 to 2017.

Tianyu Yang received the BS degree from Shanghai University, China, in June 2020, where he is currently working toward the MS degree. His research interests include linguistic steganography and natural language processing.

Hanzhou Wu (Member, IEEE) received the BS and PhD degrees from Southwest Jiaotong University, Chengdu, China, in June 2011 and June 2017, respectively. From October 2014 to October 2016, he was a visiting scholar with the New Jersey Institute of Technology, NJ, USA. He was a research staff with the Institute of Automation, Chinese Academy of Sciences, Beijing, China, from July 2017 to February 2019. He is currently an associate professor with Shanghai University, Shanghai, China. His research interests include steganography/steganalysis and digital watermarking. He has authored/co-authored around 50 research papers in peer-reviewed journals and conferences and has written three book chapters. He served the Local Organization chair of 14th IEEE International Workshop on Information Forensics and Security, the Steering Committee member of 14th/15th International Conference on Advances in Multimedia, and the guest editor for several peer-reviewed journals. He was awarded 2022 CCF-Tencent Rhino-Bird Young Faculty Open Research Fund.

Biao Yi received the BS degree from the University of South China, China, in June 2019, and the MS degree from Shanghai University, China, in June 2022. He is currently working toward the PhD degree with Nankai University, China. His research interests include natural language processing, information hiding, and artificial intelligence security.

Guorui Feng received the BS and MS degrees in computational mathematics from Jilin University, Changchun, China, in 1998 and 2001, respectively, and the PhD degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2005. From January 2006 to December 2006, he was an assistant professor with East China Normal University, Shanghai. In 2007, he was a research fellow with Nanyang Technological University, Singapore. He is currently with the School of Communication and Information Engineering, Shanghai University.

Xinpeng Zhang received the BS degree from Jilin University, China, in 1995, and the MS and PhD degrees from Shanghai University, in 2001 and 2004, respectively. Since 2004, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a full-time professor. He is also with the faculty of the School of Computer Science, Fudan University. He was with the State University of New York at Binghamton as a visiting scholar from 2010 to 2011, and also with Konstanz University as an experienced researcher, sponsored by the Alexander von Humboldt Foundation from 2011 to 2012. His research interests include multimedia security, image processing, and digital forensics. He has published more than 200 research papers. He was an associate editor of the IEEE Transactions on Information Forensics and Security from 2014 to 2017.