Discop: Provably Secure Steganography in Practice Based on “Distribution Copies”

Jinyang Ding* Kejiang Chen*✉ Yaofei Wang+ Na Zhao* Weiming Zhang*✉ Nenghai Yu*
*University of Science and Technology of China +Hefei University of Technology

Abstract—**Steganography is the act of disguising the transmission of secret information as seemingly innocent. Although provably secure steganography has been proposed for decades, it has not been mainstream in this field because its strict requirements (such as a perfect sampler and an explicit data distribution) are challenging to satisfy in traditional data environments. The popularity of deep generative models is gradually increasing and can provide an excellent opportunity to solve this problem. Several methods attempting to achieve provably secure steganography based on deep generative models have been proposed in recent years. However, they cannot achieve the expected security in practice due to unrealistic conditions, such as the balanced grouping of discrete elements and a perfect match between the message and channel distributions. In this paper, we propose a new provably secure steganography method in practice named Discop, which constructs several “distribution copies” during the generation process. At each time step of generation, the message determines from which “distribution copy” to sample. As long as the receiver agrees on some shared information with the sender, he can extract the message without error. To further improve the embedding rate, we recursively construct more “distribution copies” by creating Huffman trees. We prove that Discop can strictly maintain the original distribution so that the adversary cannot perform better than random guessing. Moreover, we conduct experiments on multiple generation tasks for diverse digital media, and the results show that Discop’s security and efficiency outperform those of previous methods.**

I. Introduction

The past few decades have witnessed massive surveillance and censorship mechanisms deployed to prevent citizens from accessing or publishing selected types of information on the Internet [1], [2]. In response, encrypted communication systems have proliferated, designed to fend off intricate censors. While these systems safeguard the confidentiality of plaintext messages, the data transmitted by them can be readily recognized as encrypted traffic, thus attracting the attention of censors. For example, encryption-based censorship circumvention techniques such as Tor [3] can be actively detected by increasingly sophisticated techniques [4]–[7] and blocked.

To cope with excessive censorship, significant efforts have been devoted to making communications look innocuous during the last few years. Among these, the most representative methods refer to information hiding technology that disguises the covert transmission of sensitive data as innocuous communication. One of the latest advances in this area is to “tunnel” secret messages in the data produced by communication applications or protocol implementations [8]–[11]. For instance, Protozoa [8] leveraged WebRTC video calls to create covert channels, where the encrypted video streams are replaced with the sensitive message. Another similar work, Balboa [9], replaced audio streams or web browsing traffic with the sensitive message. However, they rely heavily on the condition that censors can only monitor encrypted data [9], [10]. These replacement-based covert channels can be easily detected if the censor has access to plaintext data, such as controlling the WebRTC gateway that can decrypt and validate incoming media streams. Meanwhile, there is no guarantee that these channels are sustainable. Concretely, censors can systematically compromise these tools by selectively reducing the quality of encrypted channels or even blocking the delivery of encrypted traffic for which they do not have a suitable trapdoor.

To circumvent extreme censorship, steganography has been introduced into covert channels [10], [12]. Steganography is the technique that embeds secret messages in objects that closely mimic real, mundane-appearing communications so that such communications are impossible to be repressed by censors. The most familiar steganography method is least significant bit (LSB) replacement, where the sender needs an image as the cover-image, and the secret message is converted to a bitstream that replaces the LSBs of the pixel values in the cover image to obtain the stego-image. Although the visualizations of the cover-image and stego-image are similar, it is easy to detect LSB replacement from the statistical perspective [13]. To improve the covertness, several attempts have been made to minimize the modification number based on matrix encoding [14]–[16]. Considering that modifications on different regions have different impacts, researchers have shifted their focus towards *adaptive steganography*, which aims to minimize the distortion caused by embedding messages [17], [18]. However, these methods can be detected by steganalyzers [19], [20]. In recent years, deep learning-based methods for data hiding in images have been proposed, which directly utilize neural networks for message embedding and extraction, such as HiDDeN [21], SteganoGAN [22], and DeepMIH [23]. Even though involving adversarial training, they are still easily detected [23]–[25].

a) Classical provably secure steganography: Research on provably secure steganography has been carried out since two decades ago. Cachin [26] first modeled steganographic security by $D_{\mathrm{KL}}(P_{\mathrm{c}} \| P_{\mathrm{s}})$, the Kullback-Leibler divergence between the cover distribution P_{c} and the stego distribution P_{s}.

✉ Kejiang Chen and Weiming Zhang are the corresponding authors.

Hopper et al. [27] modeled steganographic security in terms of computational complexity and proposed a provably secure steganography method based on rejection sampling with poor embedding rate and excessive time consumption. Making use of the duality of steganography and source coding, Le [28] proposed a method based on arithmetic coding, which has better efficiency. However, the stringent conditions they require, such as the channel oracle and the explicit data distribution, are challenging to satisfy in the traditional environment (before the popularity of deep generative models).

b) Deep generative models and their feasibility: In the past few years, there has been a surge of deep generative models, which can approximate complex non-trivial data distributions and synthesize realistic data. Moreover, their popularity is gradually increasing, providing great scenarios and techniques for efficient, provably secure steganography in practice.

c) Efficient attempts to provably secure steganography: Thanks to deep generative models, there have been several attempts to achieve provably secure steganography in practice, such as arithmetic coding-based methods [29]–[32] and ADG [33]. However, these previous methods, in practice, would almost inevitably modify the original data distribution since the conditions they require are unrealistic, such as the balanced grouping of discrete elements and a perfect match between the message and channel distributions. They cannot achieve expected security because the adversary always has a *non-negligible advantage* in distinguishing the cover distribution P_{c} and the stego distribution P_{s}.

d) Our method: To address this problem, we propose a novel, efficient, provably secure steganography method in practice named Discop. Our insight is that during generation, we can construct multiple copies (i.e., interval assignment schemes) of a probability distribution predicted by the generative model, called "distribution copies". Concretely, we first assign each candidate token an interval of length equal to its probability within $[0, 1)$; then, we construct "distribution copies" by rotating all intervals by certain displacements. With these "distribution copies", we can express information by deciding from which "distribution copy" to sample. As long as the receiver agrees on some shared information with the sender, such as the PRNG, the seed, and the generative model, he can reconstruct the "distribution copies" and then extracts the message by determining from which "distribution copy" the current token was sampled. Since the probability of each token in different "distribution copies" is equal, the proposed scheme strictly maintains the original distribution. Thus P_{s} is strictly equal to P_{c}. A steganalyzer cannot perform better than random guessing. To further improve the embedding rate, we decompose the multivariate distribution into multiple bivariate distributions through a Huffman tree and construct "distribution copies" for each bivariate distribution recursively. In this way, more "distribution copies" are constructed.

e) Deployments: Theoretically, Discop can be deployed on any explicit generative model that can yield probability distributions. To demonstrate Discop's support for diverse generation tasks (or media), we deploy Discop on three generation tasks, including text generation, image completion, and text-to-speech.

f) Evaluation axes: We evaluate the performance of Discop on different axes inspired by the desired requirements: 1) the **security** measured by the average and maximum value of the KL divergence; 2) the **time efficiency** measured by the average time to embed one bit; 3) the **capacity efficiency** measured by the utilization of entropy, which is a new metric proposed by us and calculated by dividing the total capacity by its theoretical limit (i.e., the total entropy).

g) Contributions: The main contributions of this paper can be summarized as follows:

- **Analysis of the problems of existing attempts.** We review the existing steganography methods that attempt to achieve provable security in practice and analyze the problems preventing them from the expected security.
- **A novel steganography method.** We elaborate a novel steganography method based on "distribution copies" that share identical distribution. Furthermore, we theoretically prove its security.
- **Strategy to improve the embedding rate.** We further propose Discop, which improves the embedding rate by recursively constructing more "distribution copies".
- **Benchmarking and comparison.** We choose three typical generation tasks to deploy Discop and show that it outperforms the existing methods in terms of security and efficiency.

The source code of our implementations of Discop can be found at https://github.com/comydream/Discop.

II. Background and Related Work

A. Steganography System

Steganography is usually illustrated by Simmons' "Prisoners' Problem" [34]: Alice and Bob (steganographers) are in jail, trying to hatch an escape plan. The only way they can communicate with each other is carefully censored by the warden Eve (steganalyzer). Once Eve detects any "unusual" such as illegal words, encrypted messages, or abnormal codes, she will block their plan and throw them into high-security solitary confinement. Therefore, they must find some way to embed the secret *message* into an "innocent-looking" *cover-object* to obtain a *stego-object*.

Formally, a steganography system (stegosystem) $\Sigma_{\mathcal{D}}$ with the channel distribution $\mathcal{D}$ (alias of P_{c}) is a triple of probabilistic algorithms, $\Sigma_{\mathcal{D}} = (\mathsf{KeyGen}_{\mathcal{D}}, \mathsf{Encode}_{\mathcal{D}}, \mathsf{Decode}_{\mathcal{D}})$.

- $\mathsf{KeyGen}_{\mathcal{D}}(1^{\lambda})$ takes arbitrary input with length λ and generates a shared key $K \in \{0, 1\}^k$ with length k, which will be used in the other two algorithms.
- $\mathsf{Encode}_{\mathcal{D}}(K, \mathbf{m}, \mathcal{H})$ takes as input a shared key K, a message (i.e., the *hiddentext*) $\mathbf{m} \in \{0, 1\}^*$, and a channel history $\mathcal{H}$. It returns the *stego*, which is a symbol sequence $\mathbf{s} = s_1 \| s_2 \| \dots \| s_l$ with length l.
- $\mathsf{Decode}_{\mathcal{D}}(K, \mathbf{s}, \mathcal{H})$ takes as input a shared key K, a stego $\mathbf{s}$ and a channel history $\mathcal{H}$, and returns the message extracted from $\mathbf{s}$.

Notably, like cryptography, steganography needs to satisfy Kerckhoffs's principle [35] that Eve knows any information other than the key.

B. Definition of Steganographic Security

There are two common definitions of steganographic security. One is based on *hypothesis testing* within the framework of information theory [26] and the other is based on computational complexity theory [27], [36].

Cachin [26] first modeled steganographic security from the perspective of information theory, where given an object $\mathbf{x}$, the security of a stegosystem can be quantified by the relative entropy (a.k.a. Kullback-Leibler divergence) between the cover distribution P_{c} and the stego distribution P_{s},

$$D_{\mathrm{KL}}\left(P_{\mathrm{c}} \| P_{\mathrm{s}}\right)=\sum_{\mathbf{x} \in \mathcal{C}} P_{\mathrm{c}}(\mathbf{x}) \log \frac{P_{\mathrm{c}}(\mathbf{x})}{P_{\mathrm{s}}(\mathbf{x})},$$

which typically measures how different the two distributions are. When $D_{\mathrm{KL}}\left(P_{\mathrm{c}} \| P_{\mathrm{s}}\right)=0$, the stegosystem is considered to be *perfectly secure*. In this case, P_{c} is identical to P_{s} so that the steganalyzer cannot perform better than random guessing.

Hopper et al. [27] and Katzenbeisser and Petitcolas [36] independently proposed the complexity-theoretic definition of steganographic security, which is established by means of a probabilistic game that distinguishes the outputs of the oracle $\mathcal{O}_{\mathcal{D}}$ and $\mathsf{Encode}_{\mathcal{D}}$. The stegosystem is called secure against *chosen hiddentext attacks* if for all probabilistic polynomial time (PPT) adversaries $\mathcal{A}_{\mathcal{D}}$, it holds that

$$\left|\operatorname{Pr}\left[\mathcal{A}_{\mathcal{D}}^{\mathsf{Encode}_{\mathcal{D}}(K,\cdot,\cdot)}=1\right]-\operatorname{Pr}\left[\mathcal{A}_{\mathcal{D}}^{\mathcal{O}_{\mathcal{D}}(\cdot,\cdot)}=1\right]\right|<\mathsf{negl}(\lambda),$$

where $\mathcal{O}_{\mathcal{D}}(\cdot,\cdot)$ is an oracle that can randomly sample from the data distribution $\mathcal{D}$.

C. Classical Provably Secure Steganographic Construction

Since the definitions of steganographic security were proposed, several steganography methods were constructed, which can be divided into two categories: rejection sampling-based and arithmetic coding-based methods.

1) Rejection sampling-based methods: Assuming that there was a random sampling oracle (sampler) $\mathcal{O}_{\mathcal{D}}$ and a perfectly unbiased function f over $\mathcal{D}$ shared by the communication parties, Hopper et al. [27] proposed a provably secure steganographic construction based on rejection sampling. One encrypted message bit m_i at a time, the sender uses rejection sampling to find a symbol s_i that satisfies $f\left(s_i\right)=m_i$. The receiver, on the other hand, simply computes $m_i=f\left(s_i\right)$ for all i and concatenates them together to obtain the whole encrypted message. Von Ahn and Hopper [37] proposed public-key steganography with rejection sampling. Backes and Cachin [38] extended public-key steganography to defend against active attacks. The security of these constructions depends on the fact that rejection sampling does not destroy the channel distribution when each encrypted message bit obeys the uniform distribution $U\{0,1\}$ and f is perfectly unbiased. However, the rejection sampling-based methods suffer the following limitations in practice: 1) the expectation of time consumption grows exponentially with message length to be embedded per time step; 2) these rejection sampling algorithms fail when the distribution has very low minimum entropy. Moreover, there was a lack of an oracle that could sample objects strictly according to channel distribution in the traditional data environment.

2) Arithmetic coding-based method: Le [28] first proposed a steganographic construction based on arithmetic coding (AC), which utilizes the duality of steganography and source coding (i.e., data compression) [39]. The embedding and extraction algorithms of steganography correspond to the decompression and compression algorithms of source coding, respectively. The embedding algorithm decompresses the encrypted message into stego, and the extraction algorithm compresses the stego into the encrypted message. If the data compression scheme is perfect, the security of this construction can be reduced to that of the encryption. It has a higher capacity and better efficiency than the rejection sampling-based methods. However, in addition to the requirement of encrypting the message in advance, it also relies on a more stringent condition: explicit data distribution.

To summarize, rejection sampling-based methods require a random sampling oracle, while AC-based methods require even explicit data distribution. These conditions are challenging to satisfy in the traditional data environment (before the popularity of deep generative models) because, at that time, digital data were created by sensors or humans, whose generation processes were too complex to model.

D. Deep Generative Models and Their Feasibility

Deep learning has made great achievements in the past decade. Deep generative models, such as variational autoencoders (VAEs) [40], [41], generative adversarial networks (GANs) [42]–[45], and auto-regressive models [44], [46]–[48], have emerged and prospered. They are neural networks trained on a large amount of data and can approximate the complex non-trivial probability distribution of various types of data [42], [49], providing an excellent opportunity to provably secure steganography.

GANs and VAEs are implicit generative models that aim to build generators whose distribution is close to the real distribution of training data,

$$\mathbf{x}=G_\theta(\mathbf{z}),\ \mathbf{z} \sim p(\mathbf{z}).$$

That is, a noise vector $\mathbf{z}$ is first sampled from a prior distribution $p(\mathbf{z})$, then $\mathbf{z}$ is transformed by a neural network G_θ and finally outputs the sample $\mathbf{x}$. In the rejection sampling-based methods, we can use the generator of a GAN or VAE as a sampler for generating stego-objects that obey the data distribution.

Auto-regressive generative models, in contrast, are explicit generative models. They use the chain rule of conditional probability distribution to model the joint probability distribution of the object $\mathbf{x}$,

$$p(\mathbf{x})=\prod_{t=0}^T p\left(x_t \mid x_0, \ldots, x_{t-1}\right)=\prod_{t=0}^T p\left(x_t \mid x_{<t}\right),$$

where x_t is the basic unit of $\mathbf{x}$, like a pixel in an image, a token in a piece of text. The explicit distribution of the generated data makes efficient methods such as the AC-based method possible in practice.

The remaining problem is whether the generated data are suitable for steganography. The essence of steganography is to disguise steganographic behavior as normal behavior. Deep generative models can synthesize photo-realistic high-resolution images, luxuriant original art [43], [45], [50], [51], and articles that seem written by humans [44], [46], [47]. Currently, the generated data are widespread on the Internet. Furthermore, their popularity is gradually increasing. According to Gartner's recent research report [52], generative AI will account for 10% of all data produced by 2025. Therefore, sharing generated data can be seen as a normal behavior without raising suspicion.

E. Efficient Attempts to Provably Secure Steganography

Benefiting from the explicit generative models that can predict probability distributions of data, researchers have proposed several efficient attempts to achieve provably secure steganography in practice. These methods were dedicated to designing message embedding algorithms indistinguishable from the normal generation process, i.e., random sampling.

1) AC-based methods: Following Le's work [28], researchers have introduced deep explicit generative models to the AC-based method to make it practical, producing a series of works in recent years. We can divide them into two categories: basic AC-based methods and Meteor.

a) Basic AC-based methods: Yang et al. [29] deployed the AC-based method on the image generation task. This was the first study to apply a deep generative model to the field of provably secure steganography. Specifically, they repeatedly use an auto-regressive generative model to predict the probability distribution of the next pixel and decompress the current message fragment into a pixel using arithmetic coding. Chen et al. [30] and Ziegler et al. [31] extended the method to the text-to-speech and text generation tasks, respectively.

b) Problems of basic AC-based methods: Basic AC-based methods construct a reversible mapping between stego and message. Since the probability of a message always has the form of 2^{-l}, where $l \in \mathbb{N}$, the probability of the corresponding stego has the same form. However, a cover distribution, which the model predicts, is almost impossible to match the stego distribution perfectly. Thus, the embedding algorithm needs to tune the probability of each possible cover to fit the form of 2^{-l}, introducing a distortion. To reduce the distortion, one can lengthen the message simply by padding so that the probability of each symbol sequence becomes smaller and closer to a 2^{-l}. The distortion tends to zero as the message length approaches infinity, which is obviously impractical. Even then, the KL divergence is a negligible function with respect to the message length, not the length of the encryption key (which is the security parameter), and cannot achieve the expected security.

c) Meteor: Kaptchuk et al. [32] proposed Meteor, which addressed the problems basic AC-based methods suffer. It made two modifications to AC-based methods: 1) not to narrow the interval successively; 2) prevented the *randomness reuse* problem by re-encrypting the message each time it was used for sampling. They deployed Meteor on the text generation task.

d) Problems of Meteor: Meteor has a lower capacity than the basic AC-based methods. The reason is that the basic AC-based methods narrow the interval successively, with the generated symbol at each time step fully contributing to message embedding, while Meteor considers each generated symbol separately and cannot fully utilize the entropy.

e) Common implementation problems of AC-based methods: In their code implementations[1,2], things got worse. On each token sampled, the *original distribution* goes through "cutoff-rescale-round-remove-add" before it becomes the *actual distribution* for steganography, severely destroying the original distribution.

2) Grouping-based method: The basic idea of this method is, at each time step, to first group all tokens evenly according to their probabilities and then randomly sample a token from the group with the index corresponding to the current encrypted message fragment.

a) ADG: Zhang et al. [33] proposed a steganography method based on adaptive dynamic grouping (ADG). At each time step, they dynamically obtained the conditional probability distribution of all tokens in the vocabulary, grouped them into 2^r groups with approximately the same probability sum, and numbered them $0, 1, \ldots, 2^r - 1$. All tokens in each group represented the same message bits of length r. Then, they read r bits from the message to be embedded and converted them to a decimal number in $\{0, 1, \ldots, 2^r - 1\}$, and performed random sampling from the normalized distribution of its corresponding group to obtain the next token. They assumed that the message bits followed a uniform distribution. Since all groups had approximately the same probability sum, the probability of each token obtained by ADG sampling is close to that obtained by random sampling.

b) Problems of ADG: Theoretically, ADG can achieve perfect security if and only if the grouping is perfectly balanced. However, the problem is that since the probabilities are discrete, the requirement is almost impossible to satisfy. In most cases, the actual distribution used to embed the message is a modified distribution, which is different from the original distribution.

To facilitate understanding, we visualize the distortions introduced by the AC-based methods and ADG by giving toy examples in Appendix E.

III. Discop Methodology

As analyzed above, previous attempts to achieve provably secure steganography in practice [29], [30], [32], [33] in-

[1] https://github.com/harvardnlp/NeuralSteganography/blob/master/arithmetic.py

[2] https://gist.github.com/tusharjois/ec8603b711ff61e09167d8fef37c9b86

troduced minor damage to the original probability distribution while embedding the message. They cannot achieve the expected security because the adversary always has a *non-negligible advantage* in distinguishing the cover distributions P_c and the stego distribution P_s. To address this problem, in this section, we present a new steganography method named **Discop**, which can embed the message without destroying the original probability distribution.

A. Steganography Method Based on "Distribution Copies"

We take the text generation task as an example to describe the proposed method. Text generation leverages computational linguistics and artificial intelligence knowledge to automatically generate text that looks like human-written text. Autoregressive language models, such as GPT series models [44], [46], [47], are the most representative generative models in this area, which can predict $\mathcal{P}^{(t)} = \Pr\left[x_t \mid x_{<t}\right]$, the probability distribution of the next token given previous context $x_{<t}$.

With the predicted distribution, a strategy called *random sampling* is applied. A whole process of text generation with random sampling is usually implemented as follows. First, a pseudo-random number generator (PRNG) is used to yield a series of pseudo-random numbers $\mathbf{r} = \left\{r^{(0)}, r^{(1)}, \dots\right\}$, which follow the uniform distribution on $[0,1)$, i.e., $r^{(t)} \sim U[0,1)$. At each time step t, the generative model $\mathcal{M}$ predicts $\mathcal{P}^{(t)}$, and each token in the vocabulary V is assigned a left-closed and right-open interval in $[0,1)$ according to $\mathcal{P}^{(t)}$. Then we consume a pseudo-random number $r^{(t)}$ and select the token corresponding to the interval that $r^{(t)}$ falls into as the next token x_t, and append it to the context. This process is repeated until the termination condition is reached (e.g., the length of the generated token sequence reaches the preset maximum length).

Our insight is that during generation, we can construct multiple copies (i.e., interval assignment schemes) of the probability distribution predicted by the generative model, called "distribution copies", and use the index of "distribution copy" to express information. For example, suppose that there are only two tokens, "a" and "b" with probabilities of 0.4 and 0.6. We can assign $[0, 0.4)$ and $[0.4, 1.0)$ to "a" and "b", or assign $[0.6, 1)$ and $[0, 0.6)$ to "a" and "b". As you can see, the probability of each token is the same in several "distribution copies", meaning that the distribution of these "distribution copies" is identical. In this way, the sender can create several "distribution copies" and decide from which one to sample a token depending on the message. As long as the sender and receiver are under the same settings, including PRNG, seed, generative model, and context, they can synchronize all their states, where the seed is the number (or vector) used to initialize the PRNG and can be regarded as part of the symmetric key. Correspondingly, the receiver can extract the message by determining from which "distribution copy" the token was sampled. Notably, the existing methods use token indexes or group indexes to express information, requiring tuning the original distribution to fit the message distribution. Instead, our method uses *"distribution copy" indexes* to express information, which does not destroy the original distribution. This is the essential difference between Discop and existing methods.

There are many ways to create "distribution copies". We employ a simple implementation: rotate all intervals to the left by a certain step size, i.e., subtract this step size from all intervals, and put the part less than 0 to the far right.

A running example. For ease of understanding, here is an example. Suppose there are four tokens, namely "a", "b", "c", and "d", with probabilities of 0.1, 0.2, 0.3, and 0.4, respectively. Alice wants to embed 1 bit of information during sampling, so she needs to create $2^1 = 2$ "distribution copies", and the rotation step size is $1/2 = 0.5$. As shown in the first row in Fig. 1, she takes the order "a-b-c-d" as the initial interval assignment scheme, i.e., the "distribution copy" with index 0. Moreover, she rotates the initial interval assignment scheme to the left by 0.5 to obtain the "distribution copy" with index 1, as shown in the second row in Fig. 1. To embed message $b \in \{0, 1\}$, she samples from the "distribution copy" with the index b. Suppose she wants to embed 1, so she samples from the "distribution copy" with index 1. Suppose the pseudo-random number she consumes is $r^{(t)} = 0.2$. She finds that $r^{(t)}$ falls into the interval corresponding to "d", so she sends "d" to Bob. Bob receives "d". He can create the same "distribution copies" and consumes the same pseudo-random number $r^{(t)} = 0.2$. The received token "d" and the same "distribution copies" and consumed pseudo-random number $r^{(t)}$ allow him to uniquely determine that this "d" was obtained by sampling from the "distribution copy" with index 1, so he learns that the embedded information is 1.

The condition for unique decoding. When the tokens sampled by the consumed pseudo-random number from all "distribution copies" are *different from each other*, the receiver can uniquely decode the message. In the above example, Fig. 1 shows the "distribution copies" when 1 bit of information is to be embedded. Regardless of the value of the consumed pseudo-random number $r^{(t)}$, it satisfies that the tokens corresponding to the intervals $r^{(t)}$ falls into in all "distribution copies" are different, so Bob can uniquely decode the message. When Alice tries to embed 2 bits of information, as shown in Fig. 2, she needs to create $2^2 = 4$ "distribution copies", and the rotation step size is $1/4 = 0.25$. Note that there will be cases where the consumed pseudo-random number falls into the same token's interval in several "distribution copies". In these cases, the intervals assigned to the same token in different "distribution copies" have intersections, and the consumed pseudo-random number, unfortunately, falls into one of the intersections, resulting in the receiver cannot decode the message uniquely. We call such intersections *the disputed ranges*. For this example, to embed 1 bit of information (Fig. 1), there is no disputed range; while to embed 2 bits of information (Fig. 2), there will be disputed ranges, as shown in the parts covered by gray masks in Fig. 2. For instance, in the "distribution copies" with index 0 and 3, the intervals assigned to the token "c" are $[0.30, 0.60)$ and $[0.55, 0.85)$ respectively,

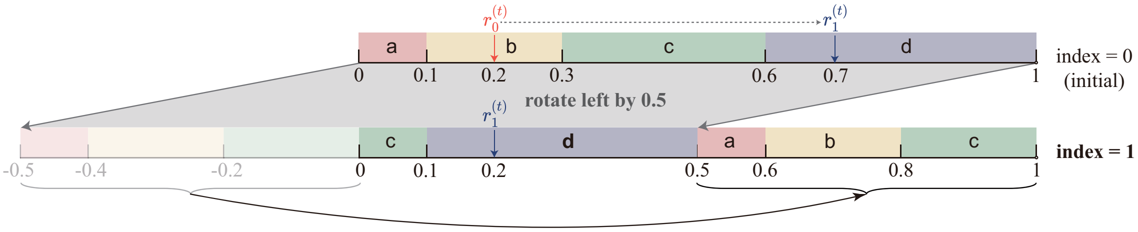


Fig. 1: **An example of the proposed steganography construction based on "distribution copies".** By rotating the interval assignment scheme (i.e., the "distribution copy" with index 0) to the left by 0.5, the "distribution copy" with index 1 is obtained. Alice wants to embed $m_i = 1$, so she samples from the "distribution copy" with index $m_i = 1$. Suppose the pseudo-random number she consumes is $r^{(t)} = 0.2$. She finds that $r^{(t)}$ falls into the interval corresponding to "d", so she sends "d". Bob receives "d" at this time step and consumes the same pseudo-random number $r^{(t)} = 0.2$. These two pieces of information make Bob uniquely determine that this "d" is obtained by sampling from the "distribution copy" with index 1, so he learns the message is $m_i = 1$. Notably, rotating the interval assignment scheme to the left by 0.5 is equivalent to rotating the consumed pseudo-random number to the right by 0.5 (e.g., $r_0^{(t)} = 0.2 \Rightarrow r_1^{(t)} = 0.7$).

and their intersection $[0.55, 0.60)$ is a disputed range. If the pseudo-random number does not fall into the disputed ranges, 2 bits of information can be embedded. Otherwise, 2 bits of information **cannot** be embedded, so we can only return to the case where only 1 bit of information can be embedded, as illustrated in Fig. 1.

Notably, the disputed ranges do not affect the correct extraction of the message. Because the receiver shares the same settings with the sender, he can calculate the disputed ranges and judge whether the pseudo-random number falls into the disputed ranges. Therefore, he can know the actual length of the embedded message and can create the same "distribution copies" to extract the message.

Embedding rate. The embedding rate is defined as the average number of bits that can be embedded per generated token. Although the disputed ranges do not affect the correct extraction of messages, they affect the embedding rate. The shorter the total length of the disputed ranges, the higher the embedding rate. We can similarly define the instantaneous embedding rate $\beta^{(t)}$ as the number of bits embedded when the t-th token is generated; then, the rotation step size at time step t is $2^{-\beta^{(t)}}$. We denote the maximum token probability at time step t as $p_{\max}^{(t)}$, and the theoretical upper bound of $\beta^{(t)}$ as $\mathrm{B}^{(t)}$. According to the condition for unique decoding, it can be deduced that $p_{\max}^{(t)}/2 < 2^{-\mathrm{B}^{(t)}} \leq p_{\max}^{(t)}$, and $\beta^{(t)} = \lfloor \mathrm{B}^{(t)} \rfloor \in \mathbb{N}$, so

$$\left\lfloor \log_2 \frac{1}{p_{\max}^{(t)}} \right\rfloor \leq \beta^{(t)} \leq \left\lceil \log_2 \frac{1}{p_{\max}^{(t)}} \right\rceil .$$

Specifically, let $\alpha = \log_2 \left(1/p_{\max}^{(t)} \right)$. If $\alpha \in \mathbb{N}$, then $\beta^{(t)} = \alpha$; otherwise, $\beta^{(t)}$ may be $\lfloor \alpha \rfloor$ or $\lceil \alpha \rceil$, and which value it is depends on whether the consumed pseudo-random number falls into the disputed ranges. We can infer that the embedding rate is asymptotic to the average of *the minimum entropy* [27] over all time steps when the generated text is long enough.

An equivalent implementation. At each time step t, rotating all intervals in $[a, b)$ is equivalent to rotating the consumed pseudo-random number $r^{(t)}$ in the opposite direction in $[a, b)$. For example, as shown in Fig. 1, rotating all intervals to the left by 0.5 is equivalent to rotating $r_0^{(t)}$ to the right by 0.5.

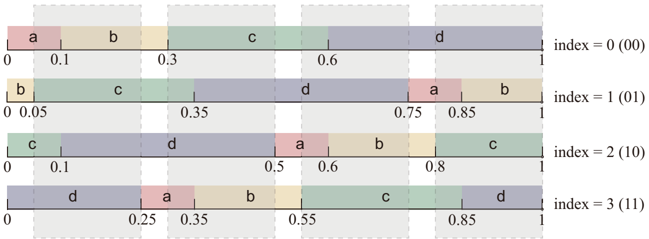


Fig. 2: To embed 2 bits of information, we need to create $2^2 = 4$ "distribution copies", and the rotation step size is $1/4 = 0.25$. The parts covered by the gray masks indicate the disputed ranges. If the consumed pseudo-random number does not fall into the disputed ranges, 2 bits of information can be embedded. Otherwise, since the unique decoding condition is not met, 2 bits of information cannot be embedded, so we can only return to the case where only 1 bit of information can be embedded, as shown in Fig. 1.

In the following, we uniformly create "distribution copies" by rotating the consumed pseudo-random numbers to the right. For convenience of description, in Algorithm 1 we define the function $\text{rotate}(a, d, e)$ to calculate the value obtained by rotating a to the right by d in the interval $[0, e)$. With the function, we start to prove the security of Discop.

Intuitive proof of security. The steganography method based on "distribution copies" performs a random sampling from one of the copies. Since the distribution of all copies is identical, the steganographic behavior does not destroy the original distribution, so $D_{\mathrm{KL}}\left(P_{\mathrm{c}} \| P_{\mathrm{s}}\right) = 0$.

Rigorous proof of security. First, consider the case where only 1 bit of information ($m_i = 0$ or 1) is embedded. Evidently, the sum of the probabilities of all possible values

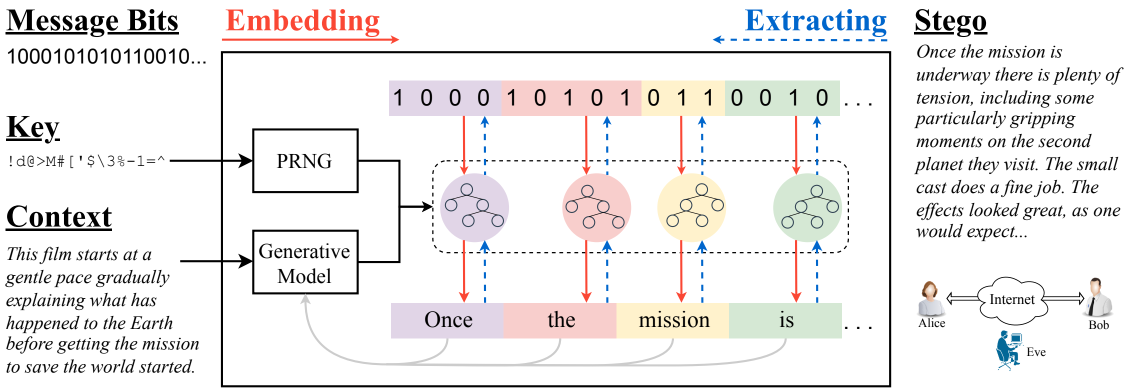

Message Bits
1000101010110010...
Embedding
Extracting
Stego
Once the mission is underway there is plenty of tension, including some particularly gripping moments on the second planet they visit. The small cast does a fine job. The effects looked great, as one would expect...
Key
!d@>M#['$\3%-1=^
PRNG
Context
This film starts at a gentle pace gradually explaining what has happened to the Earth before getting the mission to save the world started.
Generative Model
1 0 0 0 1 0 1 0 1 0 1 1 0 0 1 0 ...
Once
the
mission
is
...
Internet
Alice
Bob
Eve

Fig. 3: **An overview of Discop.** Alice and Bob need to share the same settings: 1) the initial context, 2) the PRNG, and 3) the symmetric key K (the seed for the PRNG). At each time step of Discop's embedding or extraction process, the generative model predicts the probability distribution of the next token conditioned on the current context, and a Huffman tree is created based on this distribution. During the embedding process, Alice makes a series of child node selections to obtain the next token based on 1) pseudo-random numbers generated by the PRNG and 2) the message to be embedded. She repeats the process to get the stego. Then she sends it to Bob. In the extraction process, Bob can synchronize all states with Alice and extract the message from the stego.

```
Algorithm 1: rotate$(a, d, e)$: rotate $a$ to the right by $d$ in the interval $[0, e)$
Input: Original Number $a$, Rotate Distance $d$, The
       End Point of the Interval (exclusive) $e$
Output: The Value Obtained after Rotation $b$
$b \leftarrow a + d$
if $b \geq e$ then
    $b \leftarrow b - e$
return $b$
```

is 1, i.e., $\Pr[m_i = 0] + \Pr[m_i = 1] = 1$. The pseudo-random numbers generated by the PRNG obey the uniform distribution in $[0, 1)$, that is, $r \sim U[0, 1)$. Suppose that any value a in the $[0, 1)$ interval is selected with a probability density of p, then

$$\Pr[r = a] = \Pr[r = \mathsf{rotate}(a, 0.5, 1)] = p.$$

Let r' be the equivalent value of the pseudo-random number after rotating, then

$$\begin{aligned}\Pr[r' = a] &= \Pr[r = a] \times \Pr[m_i = 0] \\ &\quad + \Pr[r = \mathsf{rotate}(a, 0.5, 1)] \times \Pr[m_i = 1] \\ &= p = \Pr[r = a].\end{aligned}$$

Similarly, this conclusion also holds when embedding multiple bits of information. In this way, we show that the proposed steganography method based on "distribution copies" does not modify the probability density of arbitrary pseudo-random numbers at each time step t, so the actual distribution for steganographic sampling is equal to the original distribution. At time step $t+1$, the history consists of steganographic tokens (generated in the previous t time steps) indistinguishable from normal ones. By analogy, the distribution at each time step is not modified, so security holds overall. Q.E.D.

B. Improving the Embedding Rate by Recursion

Although the steganography construction described above can perfectly achieve the goal of being provably secure, the embedding rate is asymptotic to the average of minimum entropy of all time steps, which is low and far from the theoretical limit (i.e., the average entropy of all time steps). Therefore, we further investigate how to improve the embedding rate.

Consider a simple case where there are three tokens, "a", "b", and "c", with probabilities of 0.5, 0.25, and 0.25. If we just use the construction in Sec. III-A, only one bit can be embedded. Our idea is that the embedding rate can be improved by grouping. Specifically, we put "b" and "c" into a group G in advance. We see that the probabilities of token "a" and group G are 0.5 and 0.5, and we can certainly embed one bit. Then if G is selected, since the normalized probabilities of "b" and "c" in G are 0.5 and 0.5, we can additionally embed one bit. Following this idea, we construct a binary tree by recursive grouping, and then embed the message bits in the process of a series of child node selections from the root node to a leaf node. This is equivalent to decomposing one-round sampling from a multivariate distribution into multiple-round sampling from bivariate distributions. It is possible to embed one bit during each sampling, thereby increasing the embedding rate.

Minimizing the total length of the disputed ranges. How to construct a binary tree so that the embedding rate is as large as possible? Similar to that discussed in Sec. III-A, at each non-leaf node, the shorter the total length of the disputed ranges is, the more likely it is to embed one bit. So our greedy strategy is to minimize the total length of the disputed ranges. We use V_j to denote the j-th token in the vocabulary V, and $p_j^{(t)}$ to denote the conditional probability of the j-th token predicted by $\mathcal{M}$ at time step t. Suppose that the groups corresponding to the left and right child nodes of each non-leaf node are G_{left} and G_{right}. It is easy to calculate the total length of the disputed ranges,

$$\left| \sum_{V_j \in G_{\text{left}}} p_j^{(t)} - \sum_{V_j \in G_{\text{right}}} p_j^{(t)} \right|.$$

The smaller the value, the higher the embedding rate. We use the Huffman tree because it can construct a series of bivariate distributions that are as balanced as possible so that this value is as small as possible.

We name the steganography method based on "**dis**tribution **cop**ies" with recursion as **Discop**. The overview of Discop is illustrated in Fig. 3. Both parties (Alice and Bob) need to share the same settings: 1) the initial context, 2) the PRNG, and 3) the symmetric key K (the seed for the PRNG). At each time step of Discop's embedding or extraction process, the generative model predicts the probability distribution of the next token conditioned on the current context, and a Huffman tree is created based on this distribution. During the embedding process, Alice makes a series of child node selections to obtain the next token based on 1) the pseudo-random numbers generated by the PRNG and 2) the message to be embedded. She repeats the process to get the stego. Then she sends it to Bob. In the extraction process, Bob can synchronize all states with Alice and extract the message from the stego. In the following, we will first introduce the two sub-procedures that are integral to both the message embedding and extraction algorithms: the construction of a Huffman tree and the selection of child nodes.

Constructing a Huffman tree. At each time step t, we create a Huffman tree [53] according to the conditional probability distribution $\mathcal{P}^{(t)}$. An example is shown in Fig. 5. It is a binary tree; each leaf node corresponds to a token, and each non-leaf node corresponds to a group containing several tokens. Appendix A provides a linear complexity algorithm for creating a Huffman tree. At the beginning of each time step, we are at the root node.

Child node selection. At each non-leaf node of the tree, we can select one of its two child nodes, possibly to express one bit. Suppose that we want to embed one bit m_i at a non-leaf node. We consume a pseudo-random number $r \in [0, \text{node.prob})$, where node.prob is the probability sum of all tokens contained in the current node. We decide which child node to select according to the interval the value $\text{rotate}(r,\ m_i \times 0.5 \times \text{node.prob},\ \text{node.prob})$ falls into. Specifically, if $m_i = 0$, we move to the child node that r falls into; otherwise, we move to the child node the value $\text{rotate}(r,\ 0.5 \times \text{node.prob},\ \text{node.prob})$ falls into. However, if both r and its rotated value fall into the interval of the same child node, that is, r falls into the disputed ranges, then we also need to move to this child node, but no bit is embedded in this child node selection, and leave m_i to the next step.

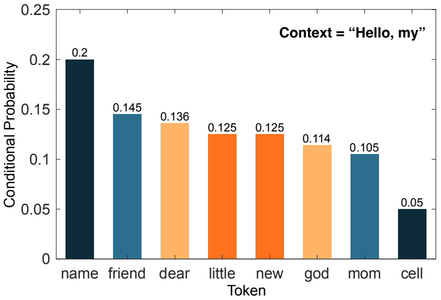


Fig. 4: An example of the distribution of the next token.

Algorithm 2: The main loop of Discop's message embedding algorithm

```
Input: Context $C$, PRNG, Language Model $\mathcal{M}$, Seed
       $K$, Message to Embed $\mathbf{m}$
Output: Stego $S$
$S \leftarrow$ " "
PRNG.set_seed$(K)$
while not the end of $\mathbf{m}$ do
    $\mathcal{P}^{(t)} \leftarrow \mathcal{M}(C)$                // predict
    $n_m, w \leftarrow \text{sample}(\mathcal{P}^{(t)}, \mathbf{m}, \text{PRNG})$
    $\mathbf{m} \leftarrow \mathbf{m}[n_m :]$
    $C \leftarrow C \| w$
    $S \leftarrow S \| w$
return $S$
```

Next, we will describe Discop's message embedding and extraction algorithms.

Embedding. The main loop of Discop's message embedding algorithm is shown in Algorithm 2. Suppose that we want to embed the message $\mathbf{m}$. At time step t, we first build a Huffman tree according to $\mathcal{P}^{(t)}$, and start from the root node. We repeat the child node selection process until the current node is already a leaf node and its corresponding token is sampled as the next token, and we append it to the end of the context. The above process is repeated until the message is completely embedded. Algorithm 3 shows the sampling process at each time step.

Extraction. The order of extracting the message is the same as that of embedding the message, i.e., traversing all tokens from the first to the last. As long as the receiver shares the same settings with the sender, all states can be synchronized. At each time step, conditioned on the same context, the receiver can use the same model to obtain the same distribution and create the same Huffman tree with the sender. According to the current received token, the path from the root node of the Huffman tree to the leaf node can be uniquely determined. In each non-leaf node on the path, the receiver can extract one bit (or none) by determining which child node the current token belongs to. Finally, he concatenates all the extracted bits together to obtain the entire message.

A running example. To make this process easy to understand,

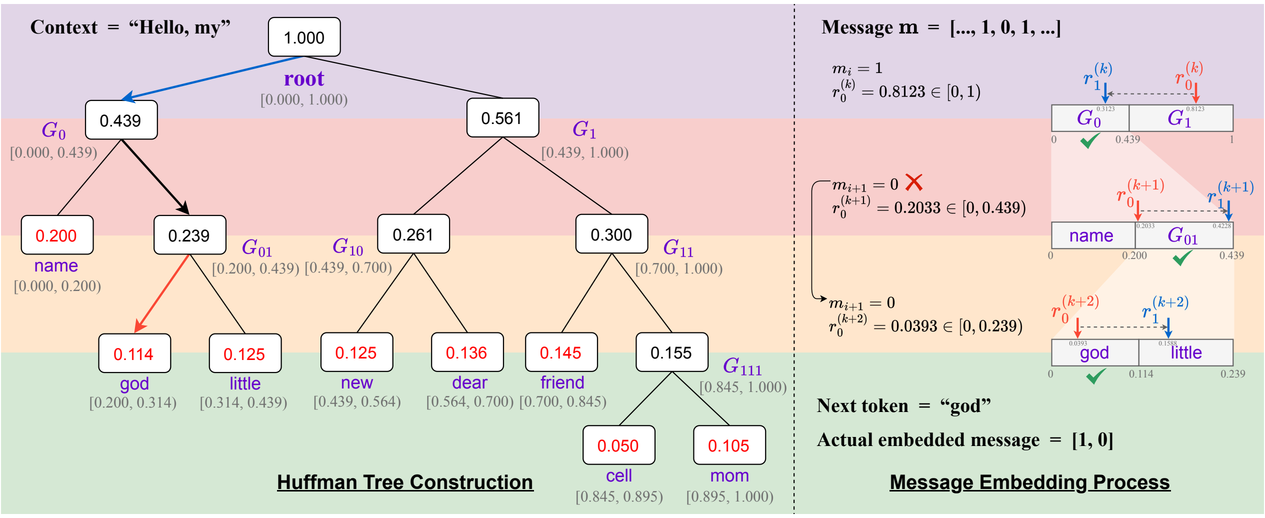


Fig. 5: An example of Discop's embedding algorithm corresponds to the distribution in Fig. 4.

here is an example. Suppose that the bits we want to embed are $[\ldots, 1,\ 0,\ 1, \ldots]$, and the context is "Hello, my". We use $\mathcal{M}$ to predict the probability distribution of the next token conditioned on the context, which is shown in Fig. 4. The Huffman tree corresponding to this distribution is shown in Fig. 5. The process of message embedding is as follows.

1) Initially, we are at the root node of the tree, which is a non-leaf node, so we make a child node selection. We use the PRNG to generate a pseudo-random number $r^{(k)} \in [0, 1)$. Assuming that we get $r^{(k)} = 0.8123$, we rotate it by 0 and 0.5, and obtain $r_0^{(k)} = 0.8123$ and $r_1^{(k)} = 0.3123$. They fall into the intervals corresponding to different child nodes, namely G_1 and G_0, so we can embed 1 bit of information $m_i = 1$ by selecting the child node G_0 corresponding to $r_1^{(k)}$.
2) Now we are at G_0, a non-leaf node, so we make a child node selection again. We use the PRNG to generate a pseudo-random number $r^{(k+1)} \in [0, 0.439)$. Assuming that we get $r^{(k+1)} = 0.2033$, we rotate it by 0 and $0.5 \times 0.439 = 0.2195$, and obtain $r_0^{(k+1)} = 0.2033$ and $r_1^{(k+1)} = 0.4228$. Unfortunately, they fall into the interval corresponding to the same child node, G_{01}, so we **cannot** embed any information here, and leave m_{i+1} to the next step. Since both $r_0^{(k+1)}$ and $r_1^{(k+1)}$ fall into G_{01}, we have to select G_{01}.
3) Now we are at G_{01}, a non-leaf node, so we make a child node selection again. We use the PRNG to generate a pseudo-random number $r^{(k+2)} \in [0, 0.239)$. Assuming that we get $r^{(k+2)} = 0.0393$, we rotate it by 0 and $0.5 \times 0.239 = 0.1195$, and obtain $r_0^{(k+2)} = 0.0393$ and $r_1^{(k+2)} = 0.1588$. They fall into the intervals corresponding to different child nodes, namely the token "god" and the token "little", so we can embed 1 bit of information $m_{i+1} = 0$ by selecting the token "god" corresponding to $r_0^{(k+2)}$.
4) Now we are at a leaf node, so we sample the token "god" corresponding to this node as the next token and append it to the context. After the above process, we actually embed 2 bits of information $[1, 0]$. The subsequent bits will be embedded by similar generation processes later.

We note that a previous Huffman coding-based steganography method [54] also involved Huffman trees. However, it used the token indexes to express the message, similar to the AC-based methods. It would damage the original distribution more seriously than the AC-based methods. Instead, Discop uses the "distribution copy" indexes to express the message without destroying the distribution.

Complexity. At each time step, the Discop algorithm mainly includes two processes: creating a Huffman tree and a series of child node selections (walking from the root node to the leaf node). Let the size of the vocabulary be $|V|$. Creating a Huffman tree can be implemented in linear complexity, i.e. $O(|V|)$. The average complexity of walking from the root node to a leaf node of the tree is $O(\log |V|)$.

IV. Deployments

Theoretically, Discop can be deployed on any explicit generative model that can yield probability distributions. Here, to illustrate Discop's support for diverse generation tasks (or media), we select several typical generation tasks and corresponding publicly available pre-trained models to deploy Discop on, as shown in Table I. We have described in detail how to deploy Discop on the text generation task in Sec. III. Similarly, we also deploy Discop on auto-regressive models for image completion and text-to-speech tasks. Note that the deployment scenarios of Discop are not limited to these.

Before starting covert communication, the two parties must agree on a protocol, i.e., they should share the same settings. The basic settings include the steganography method, the PRNG, and the seed. In addition, there may be additional

Algorithm 3: sample$(\mathcal{P}^{(t)}, \mathbf{m}, \mathrm{PRNG})$: sampling at time step t

```
Input: Distribution $\mathcal{P}^{(t)}$, Message to Embed $\mathbf{m}$,
       PRNG
Output: Number of Bits Embedded $n_m$, Selected Next
        Token $w$
struct {
    token, prob, left, right
} Node
node ← create_tree $(\mathcal{P}^{(t)})$    // the root node
$n_m \leftarrow 0$
while not node.is_leaf() do
    $r \leftarrow$ PRNG.random $(0, \mathrm{node.prob})$
      // $r \in [0, \mathrm{node.prob})$
    $r_0 \leftarrow r$
    $r_1 \leftarrow$ rotate $(r,\ 0.5 \times \mathrm{node.prob},\ \mathrm{node.prob})$
    separator ← node.left.prob
    next ← [ ]
    if $r_0 <$ separator then
        next[0] ← node.left
    else
        next[0] ← node.right
    if $r_1 <$ separator then
        next[1] ← node.left
    else
        next[1] ← node.right
    if next[0] ≠ next[1] then   // embed one bit
        $n_m \leftarrow n_m + 1$
    node ← next $[\mathbf{m}[n_m]]$
$w \leftarrow$ node.token
return $n_m$, $w$
```

settings for a specific task, which will be described separately in the following.

A. Text Generation

To deploy Discop on this task, both communication parties need to share the same initial context, which can be empty (for unconditional generation) or consist of a specified number of tokens or sentences. As shown in Fig. 6(a), for example, if both parties agree to use n tokens as the initial context, the sender can choose arbitrary n tokens as the initial context, then use the Discop embedding algorithm to write a continuation of it with the message embedded to obtain the stego-text, and send the concatenation of the initial context and stego-text to the receiver. After receiving the text, the receiver can split it into two parts according to the protocol and then use the Discop extraction algorithm to extract the message.

B. Image Completion

To deploy Discop on this task, both parties should share the initial context of the image, which can be achieved by sharing a *context proportion parameter* p_c (which indicates the proportion of the initial context to the full image). As shown in Fig. 6(b), the sender uses the Discop embedding algorithm to complete an image part of size $(p_c \times m) \times n$ to a complete image of size $m \times n$ in a pixel-by-pixel manner and sends it to the receiver. The receiver knows p_c and can segment the received image into two parts: the initial context and the stego-image, then use the Discop extraction algorithm to extract the message.

C. Text-to-Speech

To deploy Discop on this task, as shown in Fig. 6(c), the sender uses the Discop embedding algorithm to convert the text to a corresponding speech in a sample-by-sample[3] manner. The receiver uses the Discop extraction algorithm to extract the message. Note that the receiver should share the text corresponding to the speech with the sender, which can be achieved by the sender sending the text directly along with the speech or by the receiver performing speech recognition.

For GPT-2 [47], DistilGPT-2 [55], Transformer-XL [56], and Image GPT [57], we directly employ the pre-trained models provided by Hugging Face [60]. For Tacotron [58] and WaveRNN [59], we employ the public pre-trained models from GitHub[4,5].

V. Experiments and Evaluation

In this section, we conduct experiments to present the performance of Discop mainly in terms of security and efficiency, and compare Discop with previous methods attempting to achieve provable security in practice.

A. Setup

We follow the deployments described in Sec. IV to carry out experiments. In the experiments, we introduce *nucleus sampling* [61], a.k.a. top-p sampling, which is now widely used in generation tasks. It chooses the smallest possible set whose cumulative probability exceeds p and then normalizes the probability distribution of this set and performs random sampling from the normalized distribution. To better evaluate the performance in various scenarios, we set the truncation parameter $p = 0.80,\ 0.92,\ 0.95,\ 0.98,\ 1.00$. Other settings on the three deployed tasks are as follows.

1) Text generation: For each p, we select 100 pieces of text from the IMDb dataset [62]. We use the first three sentences of each sample as the context and generate the subsequent 100 tokens conditioned on the context.

2) Image completion: For each p, we select 100 images from the CelebA dataset [63]. We resize each image to the size specified by the model and provide the upper part of the image to the model to complete the image.

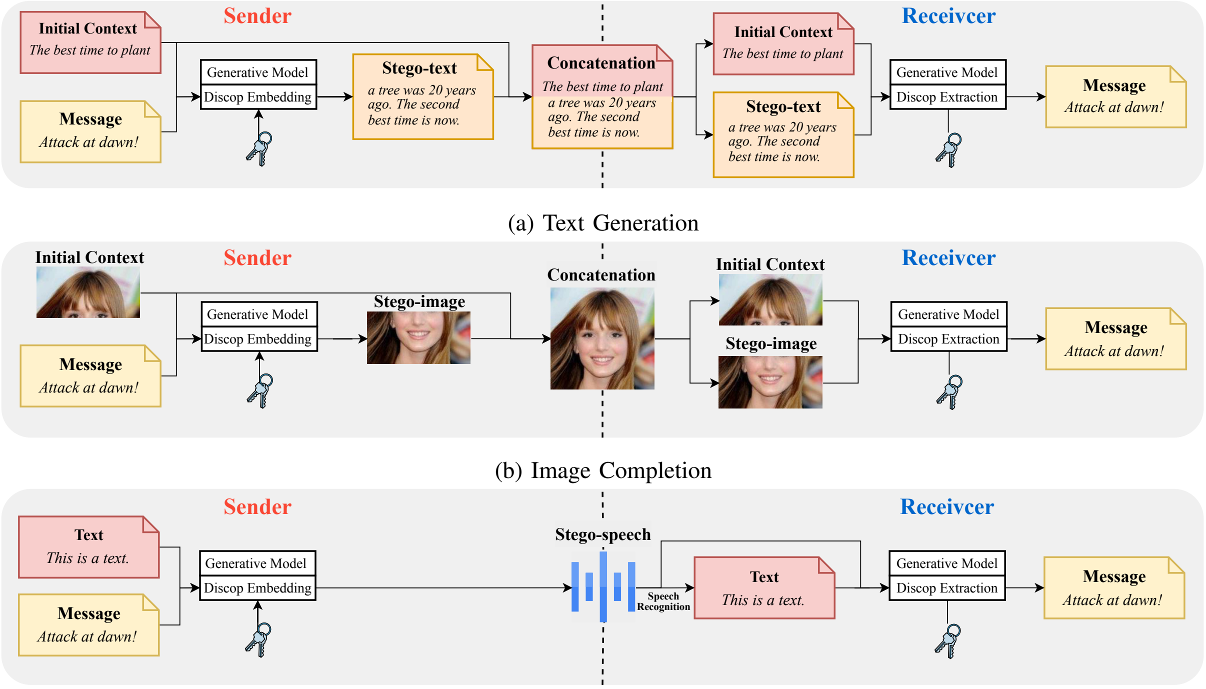
[3] The "sample" here refers to an element in a waveform.

[4] https://github.com/bshall/Tacotron

[5] https://github.com/bshall/UniversalVocoding

TABLE I: The tasks/models we implemented the deployment of Discop on.

Task	Description	Model
Text Generation	Given a context, write a continuation of it.	GPT-2 [47], DistilGPT-2 [55], Transformer-XL [56]
Image Completion	Given part of an image (e.g., the upper half part), complete it.	Image GPT [57]
Text-to-Speech	Given a piece of text, synthesize its corresponding speech.	Tacotron [58] + WaveRNN [59]



(a) Text Generation

(b) Image Completion

(c) Text-to-Speech

Fig. 6: Schematic diagrams of deploying Discop on the three tasks.

3) Text-to-speech (TTS): For each p, we select 100 pieces of text from the IMDb dataset [62]. We use the model to generate the speech corresponding to the first sentence of each text.

We compare Discop with Meteor [32] and ADG [33] only on the text generation task because both of them are originally deployed only on this task. The base version of Meteor has a low embedding rate, for which its authors have designed a heuristic sorting algorithm to improve the embedding rate. We benchmark Meteor for both the base version and the version with sorting. Since Meteor is an enhanced version of the basic AC-based method, we do not need to benchmark the basic AC-based method.

All experiments are carried out under the same hardware settings (CPU 3.00GHz, 128GB RAM, and NVIDIA RTX 3090). The involved datasets are loaded from the Datasets library provided by Hugging Face [64].

B. Metrics

We evaluate the performance of Discop in terms of both security and efficiency. We consider that the efficiency of steganography mainly includes two aspects: the embedding rate and the time consumption. To better characterize the distance between the embedding rate and the theoretical limit (i.e., the average value of entropy for all time steps), we propose a new metric, the utilization rate of entropy.

1) Security: Again, the primary pursuit of steganography is “behavioral security”. That is, steganographic behavior should be indistinguishable from normal behavior. Here, we pursue that the generated text with message embedded (stegos) is indistinguishable from the “normal” generated text without message embedded (covers). We follow Cachin’s formalization of steganographic security from the perspective of information theory [26] and use the KL divergence between P_{c} and P_{s}, i.e., $D_{\mathrm{KL}}\left(P_{\mathrm{c}} \| P_{\mathrm{s}}\right)$, to measure security, which is also the foremost metric to measure the competence of a steganography method. During each token sampling, the model gives the *original distribution*. To embed message bits, a steganography method may modify the original distribution (e.g., ADG tunes the total probability of each group to $2^{-n}, n \in \mathbb{N}$), and we can get the *actual distribution* for steganographic sampling. So the KL divergence between these two distributions can be calculated. In the experiment, we use two KL divergence metrics:

- **Ave KLD:** The average value of the KL divergence for all time steps. It is obtained by dividing the cumulative

TABLE II: Quantitative comparison with the previous attempts using GPT-2.

Method	p	Total Time (seconds)	Ave Time ↓ (seconds/bit)	Ave KLD ↓ (bits/token)	Max KLD ↓ (bits/token)	Capacity (bits/token)	Entropy (bits/token)	Utilization ↑
ADG	0.80	96.71	3.16E-03	7.93E-03	6.76E-02	3.07	3.95	0.78
	0.92	104.48	2.57E-03	1.02E-02	4.75E-02	4.06	4.93	0.82
	0.95	114.72	2.62E-03	1.09E-02	4.73E-02	4.38	5.34	0.82
	0.98	150.68	3.08E-03	1.20E-02	4.54E-02	4.89	5.83	0.84
	1.00	846.27	1.57E-02	1.31E-02	4.99E-02	5.39	6.26	0.86
Meteor w/o sort	0.80	95.58	3.73E-03	5.13E-02	8.28E+00	2.56	3.83	0.67
	0.92	96.16	2.79E-03	8.17E-03	5.62E+00	3.44	4.82	0.71
	0.95	98.57	2.61E-03	3.40E-03	1.30E+00	3.78	5.15	0.73
	0.98	105.37	2.51E-03	6.59E-04	1.74E+00	4.20	5.61	0.75
	1.00	251.48	**5.56E-03**	1.05E-06	1.68E-05	4.52	5.96	0.76
Meteor	0.80	282.18	9.71E-03	5.57E-02	9.01E+00	2.91	3.76	0.77
	0.92	1359.87	3.33E-02	9.34E-03	4.63E+00	4.09	4.87	0.84
	0.95	2334.54	5.21E-02	2.77E-03	6.98E-01	4.48	5.23	0.86
	0.98	5559.88	1.16E-01	5.57E-04	8.23E-01	4.79	5.60	0.86
	1.00	47301.20	9.11E-01	1.06E-06	1.68E-05	5.19	5.98	0.87
Discop w/o recursion (Proposed)	0.80	101.33	5.52E-03	**0**	**0**	1.84	3.84	0.48
	0.92	102.78	5.00E-03	**0**	**0**	2.06	4.83	0.43
	0.95	103.11	4.74E-03	**0**	**0**	2.17	5.29	0.41
	0.98	105.81	4.70E-03	**0**	**0**	2.25	5.68	0.40
	1.00	145.81	6.38E-03	**0**	**0**	2.29	6.03	0.38
Discop (Proposed)	0.80	104.30	**2.99E-03**	**0**	**0**	3.48	3.79	**0.92**
	0.92	104.36	**2.29E-03**	**0**	**0**	4.55	4.86	**0.94**
	0.95	107.07	**2.21E-03**	**0**	**0**	4.84	5.18	**0.94**
	0.98	115.13	**2.17E-03**	**0**	**0**	5.29	5.59	**0.95**
	1.00	362.63	6.29E-03	**0**	**0**	5.76	6.08	**0.95**
Random Sampling	0.80	91.21	N/A	0	0	N/A	3.80	N/A
	0.92	90.89	N/A	0	0	N/A	4.80	N/A
	0.95	92.39	N/A	0	0	N/A	5.15	N/A
	0.98	95.20	N/A	0	0	N/A	5.59	N/A
	1.00	174.09	N/A	0	0	N/A	5.87	N/A

KL divergence over all time steps by the total number of tokens, indicating the average degree of damage to the original distribution by the steganography method. The lower, the better.

- **Max KLD:** The maximum value of the KL divergence for all time steps. This indicates the most severe degree of damage to the original distribution by the steganography method. The lower, the better.

2) Utilization: The embedding rate, a.k.a. capacity in some papers, is typically defined as the average number of bits that can be embedded per generated token. However, random sampling is used in the generation process so that the theoretical limit of the length of the message that can be embedded (i.e., the sum of entropy over all time steps) is not fixed. Therefore, to better evaluate the embedding ability, we propose a new metric, the utilization rate of entropy, **Utilization** for short, which is defined as the ratio of the total length of the embedded message to the entropy sum over all time steps. The higher, the better.

3) Time: To realize real-time covert communication, we expect to complete steganographic embedding and extraction in a relatively short time. We define a metric called **Ave Time**, which is obtained by dividing the total time of the entire process by the total length of the embedded message. The shorter, the better. In addition, we expect that the additional time consumption caused by Discop's embedding is as little as possible compared to the normal generation (random sampling) process. We performed timing tests on these models for the Discop generation process and the random sampling process.

In addition, we conduct steganalysis and generation quality evaluation experiments for Discop in Appendix C and D.

C. Results and Analysis

Note that for the text generation task, we only show the experimental results of deploying these methods on GPT-2 because the results obtained by deploying these methods on the other two models are similar.

1) Comparison to Baselines: We compare Discop with Meteor [32] and ADG [33] only on the text generation task because both of them are deployed only on this task. The experimental results are presented in Table II. In this table, "Meteor w/o sort" and "Meteor" correspond to the base version and the version with sorting of Meteor, respectively. "Discop w/o recursion" corresponds to our elementary method proposed in Sec. III-A. "Random Sampling" is a reference that indicates the "normal" generation without embedding any message. According to the previous definitions and analysis, **Ave Time**, **Ave KLD**, **Max KLD**, and **Utilization** are the key metrics to measure the performance of steganography methods, and **Total Time**, **Embedding Rate**, and **Entropy** are also listed in the table for reference. Overall, our method outperformed the other methods in almost all metrics. The experimental results are described and analyzed as follows.

a) Security: The KL divergence of Meteor and ADG is far from negligible, which may cause the adversary to gain a non-negligible advantage. The non-negligible KL divergence in ADG is likely due to issues around grouping discrete probabilities (as discussed in Sec. II-E(2)(b)), while in Meteor it is likely due to implementation problems (as discussed in Sec. II-E(1)(e)). Since Discop does not fundamentally change the original distributions, it achieves the goal that the KL divergence is zero, which is not achieved by other methods.

b) Utilization: The utilization of entropy by ADG and Meteor is in the range of 0.77 to 0.87, while that by Discop is in the range of 0.92 to 0.95. Discop makes better use of entropy than other methods, and the recursion strategy significantly improves the utilization. For the embedding rate, Discop can embed 3.48 to 5.76 bits of information per generated token, which is better than baselines.

c) Time: Under most truncation settings, all methods except Meteor can embed the messages at a competitively low time consumption. Compared to "normal" random sampling, they do not take much extra time. In contrast, Meteor adopts a heuristic interval sorting algorithm, which results in enormous time costs. As a result, to embed a message of the same length, Meteor takes 3.25× to 144.88× the time cost of Discop, which is probably unacceptable when real-time covert communication is needed.

2) Evaluation of additional time consumption: The experimental results are shown in Table III. Compared to random sampling, most of the additional time of Discop's embedding algorithm is consumed in creating Huffman trees, which is proportional to the number of candidate tokens. As can be seen, for GPT-2 with 50,257 candidate tokens, the top-p truncation ($p \neq 1.00$) removes a large number of candidate tokens, making the additional time very short. For Image GPT with 512 candidate tokens and WaveRNN with 1024 candidate tokens, the number of candidate tokens is much smaller than that of GPT-2, so regardless of the value of p, the extra time introduced by Discop's message embedding algorithm is relatively short.

VI. Discussion

A. Out-of-Band Cost

In general, the sender needs to use an *encryption key* to encrypt the message before performing the embedding algorithm; on the other hand, the receiver needs to use the same key to decrypt the message after performing the extraction algorithm. Therefore, steganography involves two keys: the encryption key and the steganography key, which need to be transmitted over an out-of-band channel in advance.

For Discop, the steganographic key is the seed used to initialize the PRNG. Other important components including the generative model and the PRNG—which are publicly available—can be considered part of the protocol.

TABLE III: The additional time consumption introduced by Discop to embed messages.

Task/Model	p	Random Sampling Time (seconds)	Discop Time (seconds)	Ratio
Text Generation GPT-2	0.80	91.21	104.30	1.14
	0.92	90.89	104.36	1.15
	0.95	92.39	107.07	1.16
	0.98	95.20	115.13	1.21
	1.00	174.09	362.63	2.08
Image Completion Image GPT	0.80	739.82	741.35	1.00
	0.92	740.57	750.96	1.01
	0.95	742.79	832.52	1.12
	0.98	738.57	763.67	1.03
	1.00	752.24	759.24	1.01
Text-to-speech WaveRNN	0.80	2500.56	2679.60	1.07
	0.92	2522.93	2599.81	1.03
	0.95	2520.88	2649.25	1.05
	0.98	2537.15	2700.39	1.06
	1.00	2744.50	3582.87	1.31

To avoid the out-of-band cost, a possible solution is introducing an existing public-key steganography method such as [37]. Similar to hybrid encryption, a public-key steganography session could establish a shared state, and Discop can use that state for its operation.

B. Limitations

We would like to clarify the limitations of Discop.

- Discop is only suitable for explicit generative models that can yield probability distributions, e.g., auto-regressive models.
- Discop requires both parties to share the pseudo-random numbers used to perform sampling, so it can only construct symmetric-key steganography but not public-key steganography.

VII. Conclusion

In this paper, we analyze the problems faced by previous attempts to achieve provably secure steganography in practice. To overcome these problems, we present Discop, a novel, efficient, provably secure steganography method in practice based on "distribution copies", which does not destroy the original data distribution when embedding the message. We deploy Discop on diverse generation tasks (or media) and conduct a series of experiments. The experimental results show that Discop's security and efficiency outperform previous attempts. Moreover, the embedding rate of Discop reaches approximately 0.92 to 0.95 of the theoretical limit.

Acknowledgement

We thank the reviewers for their valuable comments. This work was supported in part by the Natural Science Foundation of China under Grant 62102386, 62002334, 62072421, 62121002 and U20B2047, by Xiaomi Young Scholars Program, and by Open Fund of Anhui Province Key Laboratory of Cyberspace Security Situation Awareness and Evaluation.

References

[1] F. Cramer, "Hiding in plain sight. amy sou wu's the kandinsky collective," *Inmaterial. Diseño, Arte y Sociedad*, vol. 2, no. 4, pp. 105–114, 2017.

[2] S. Landau, "Highlights from making sense of snowden, part ii: What's significant in the nsa revelations," *IEEE Security & Privacy*, vol. 12, no. 1, pp. 62–64, 2014.

[3] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, M. Blaze, Ed. USENIX, 2004, pp. 303–320. [Online]. Available: http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html

[4] M. W. Al Nabki, E. Fidalgo, E. Alegre, and I. de Paz, "Classifying Illegal Activities on Tor Network Based on Web Textual Contents," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, 2017, pp. 35–43.

[5] A. Cuzzocrea, F. Martinelli, F. Mercaldo, and G. Vercelli, "Tor traffic analysis and detection via machine learning techniques," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 4474–4480.

[6] F. A. Saputra, I. U. Nadhori, and B. F. Barry, "Detecting and blocking onion router traffic using deep packet inspection," in *2016 International Electronics Symposium (IES)*, Sep. 2016, pp. 283–288.

[7] D. Sarkar, P. Vinod, and S. Y. Yerima, "Detection of Tor Traffic using Deep Learning," in *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, 2020, pp. 1–8.

[8] D. Barradas, N. Santos, L. Rodrigues, and V. Nunes, "Poking a Hole in the Wall: Efficient Censorship-Resistant Internet Communications by Parasitizing on WebRTC," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 35–48.

[9] M. B. Rosen, J. Parker, and A. J. Malozemoff, "Balboa: Bobbing and Weaving around Network Censorship," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3399–3413.

[10] G. Figueira, D. Barradas, and N. Santos, "Stegozoa: Enhancing webrtc covert channels with video steganography for internet censorship circumvention," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 1154–1167.

[11] J. K. H. Iv, M. Georgiou, A. J. Malozemoff, and T. Shrimpton, "Security Foundations for Application-Based Covert Communication Channels," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 1971–1986.

[12] J. Mahmod and M. Hicks, "Invisible bits: hiding secret messages in sram's analog domain," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 1086–1098.

[13] A. D. Ker, "Improved detection of lsb steganography in grayscale images," in *International workshop on information hiding*. Springer, 2004, pp. 97–115.

[14] J. Fridrich, "Minimizing the embedding impact in steganography," in *Proceedings of the 8th Workshop on Multimedia and Security*, 2006, pp. 2–10.

[15] X. Zhang and S. Wang, "Efficient steganographic embedding by exploiting modification direction," *IEEE Communications letters*, vol. 10, no. 11, pp. 781–783, 2006.

[16] W. Zhang and X. Wang, "Generalization of the zzw embedding construction for steganography," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 3, pp. 564–569, 2009.

[17] T. Filler, J. Judas, and J. Fridrich, "Minimizing additive distortion in steganography using syndrome-trellis codes," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 920–935, 2011.

[18] W. Li, W. Zhang, L. Li, H. Zhou, and N. Yu, "Designing near-optimal steganographic codes in practice based on polar codes," *IEEE Transactions on Communications*, vol. 68, no. 7, pp. 3948–3962, 2020.

[19] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Transactions on information Forensics and Security*, vol. 7, no. 3, pp. 868–882, 2012.

[20] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1181–1193, 2018.

[21] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, "Hidden: Hiding data with deep networks," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, 2018, pp. 682–697. [Online]. Available: https://doi.org/10.1007/978-3-030-01267-0_40

[22] K. A. Zhang, A. Cuesta-Infante, L. Xu, and K. Veeramachaneni, "Steganogan: High capacity image steganography with gans," *arXiv preprint arXiv:1901.03892*, 2019.

[23] Z. Guan, J. Jing, X. Deng, M. Xu, L. Jiang, Z. Zhang, and Y. Li, "Deepmih: Deep invertible network for multiple image hiding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[24] J. Tan, X. Liao, J. Liu, Y. Cao, and H. Jiang, "Channel attention image steganography with generative adversarial networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 2, pp. 888–903, 2022. [Online]. Available: https://doi.org/10.1109/TNSE.2021.3139671

[25] S. Baluja, "Hiding images within images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, pp. 1685–1697, 2020. [Online]. Available: https://doi.org/10.1109/TPAMI.2019.2901877

[26] C. Cachin, "An Information-Theoretic Model for Steganography," in *Information Hiding*, D. Aucsmith, Ed. Berlin, Heidelberg: Springer, 1998, pp. 306–318.

[27] N. J. Hopper, J. Langford, and L. von Ahn, "Provably Secure Steganography," in *Advances in Cryptology — CRYPTO 2002*, M. Yung, Ed. Berlin, Heidelberg: Springer, 2002, pp. 77–92.

[28] T. V. Le, "Efficient provably secure public key steganography," *IACR Cryptol. ePrint Arch.*, p. 156, 2003. [Online]. Available: http://eprint.iacr.org/2003/156

[29] K. Yang, K. Chen, W. Zhang, and N. Yu, "Provably secure generative steganography based on autoregressive model," in *Digital Forensics and Watermarking - 17th International Workshop, IWDW 2018, Jeju Island, Korea, October 22-24, 2018, Proceedings*, ser. Lecture Notes in Computer Science, C. D. Yoo, Y. Q. Shi, H. Kim, A. Piva, and G. Kim, Eds., vol. 11378. Springer, 2018, pp. 55–68. [Online]. Available: https://doi.org/10.1007/978-3-030-11389-6_5

[30] K. Chen, H. Zhou, H. Zhao, D. Chen, W. Zhang, and N. Yu, "Distribution-preserving steganography based on text-to-speech generative models," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[31] Z. Ziegler, Y. Deng, and A. Rush, "Neural Linguistic Steganography," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, pp. 1210–1215.

[32] G. Kaptchuk, T. M. Jois, M. Green, and A. D. Rubin, "Meteor: Cryptographically Secure Steganography for Realistic Distributions," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event Republic of Korea: ACM, Nov. 2021, pp. 1529–1548.

[33] S. Zhang, Z. Yang, J. Yang, and Y. Huang, "Provably Secure Generative Linguistic Steganography," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, 2021, pp. 3046–3055.

[34] G. J. Simmons, "The prisoners' problem and the subliminal channel," in *Advances in Cryptology*. Springer, 1984, pp. 51–67.

[35] A. Kerckhoffs, "La cryptographic militaire," *Journal des sciences militaires*, pp. 5–38, 1883.

[36] S. Katzenbeisser and F. A. P. Petitcolas, "Defining security in steganographic systems," in *Security and Watermarking of Multimedia Contents IV*, E. J. Delp III and P. W. Wong, Eds., San Jose, CA, Apr. 2002, pp. 50–56. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=877726

[37] L. v. Ahn and N. J. Hopper, "Public-key steganography," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2004, pp. 323–341.

[38] M. Backes and C. Cachin, "Public-Key Steganography with Active Attacks," in *Theory of Cryptography*, ser. Lecture Notes in Computer Science, J. Kilian, Ed. Berlin, Heidelberg: Springer, 2005, pp. 210–226.

[39] R. Barron, B. Chen, and G. Wornell, "The duality between information embedding and source coding with side information and some applications," *IEEE Transactions on Information Theory*, vol. 49, no. 5, pp. 1159–1180, 2003.

[40] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *Proceedings of the 2th International Conference on Learning Representations*, Banff, AB, Canada, 2014.

[41] D. P. Kingma, M. Welling *et al.*, “An introduction to variational autoencoders,” *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[42] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets,” in *Advances in Neural Information Processing Systems*, vol. 27. Cambridge, MA, USA: Curran Associates, Inc., 2014, pp. 2672–2680.

[43] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2223–2232.

[44] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language Models are Few-Shot Learners,” in *Advances in Neural Information Processing Systems*, vol. 33. Online: Curran Associates, Inc., 2020, pp. 1877–1901.

[45] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski, “StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Montreal, QC, Canada: IEEE, 2021, pp. 2085–2094.

[46] A. Radford and K. Narasimhan, “Improving Language Understanding by Generative Pre-Training,” 2018.

[47] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners,” p. 24, 2019.

[48] A. van den Oord, N. Kalchbrenner, L. Espeholt, k. kavukcuoglu, O. Vinyals, and A. Graves, “Conditional Image Generation with PixelCNN Decoders,” in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.

[49] L. Ruthotto and E. Haber, “An Introduction to Deep Generative Modeling,” *arXiv:2103.05180 [cs]*, Apr. 2021.

[50] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, “Zero-Shot Text-to-Image Generation,” in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, Jul. 2021, pp. 8821–8831.

[51] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical Text-Conditional Image Generation with CLIP Latents,” *arXiv:2204.06125 [cs]*, Apr. 2022.

[52] Gartner, “Top strategic technology trends for 2022,” *Gartner*, 2022.

[53] D. A. Huffman, “A Method for the Construction of Minimum-Redundancy Codes,” *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[54] Z. Yang, X. Guo, Z. Chen, Y. Huang, and Y. Zhang, “Rnn-stega: Linguistic steganography based on recurrent neural networks,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1280–1295, 2019. [Online]. Available: https://doi.org/10.1109/TIFS.2018.2871746

[55] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” vol. abs/1910.01108, 2019.

[56] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, “Transformer-XL: Attentive Language Models beyond a Fixed-Length Context,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 2978–2988.

[57] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, “Generative Pretraining From Pixels,” in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 1691–1703.

[58] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, “Tacotron: Towards End-to-End Speech Synthesis,” in *Interspeech 2017*. ISCA, Aug. 2017, pp. 4006–4010.

[59] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Oord, S. Dieleman, and K. Kavukcuoglu, “Efficient Neural Audio Synthesis,” in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 2410–2419.

[60] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, “Transformers: State-of-the-Art Natural Language Processing,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, 2020, pp. 38–45.

[61] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, “The Curious Case of Neural Text Degeneration,” in *8th International Conference on Learning Representations*. Addis Ababa, Ethiopia,: OpenReview.net, Apr. 2020.

[62] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[63] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep Learning Face Attributes in the Wild,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3730–3738.

[64] Q. Lhoest, A. Villanova del Moral, Y. Jernite, A. Thakur, P. von Platen, S. Patil, J. Chaumond, M. Drame, J. Plu, L. Tunstall, J. Davison, M. Šaško, G. Chhablani, B. Malik, S. Brandeis, T. Le Scao, V. Sanh, C. Xu, N. Patry, A. McMillan-Major, P. Schmid, S. Gugger, C. Delangue, T. Matussière, L. Debut, S. Bekman, P. Cistac, T. Goehringer, V. Mustar, F. Lagunas, A. Rush, and T. Wolf, “Datasets: A community library for natural language processing,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 175–184. [Online]. Available: https://aclanthology.org/2021.emnlp-demo.21

[65] 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.

[66] Y. Niu, J. Wen, P. Zhong, and Y. Xue, “A Hybrid R-BILSTM-C Neural Network Based Text Steganalysis,” *IEEE Signal Processing Letters*, vol. 26, no. 12, pp. 1907–1911, Dec. 2019.

[67] H. Yang, Y. Bao, Z. Yang, S. Liu, Y. Huang, and S. Jiao, “Linguistic Steganalysis via Densely Connected LSTM with Feature Pyramid,” in *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*. Denver CO USA: ACM, Jun. 2020, pp. 5–10.

[68] W. Luo, H. Li, Q. Yan, R. Yang, and J. Huang, “Improved audio steganalytic feature and its applications in audio forensics,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 2, pp. 1–14, 2018.

[69] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 6626–6637. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html

[70] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, “Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2001, 7-11 May, 2001, Salt Palace Convention Center, Salt Lake City, Utah, USA, Proceedings*, 2001, pp. 749–752. [Online]. Available: https://doi.org/10.1109/ICASSP.2001.941023

Appendix

A. Linear Complexity Algorithm for Creating a Huffman Tree

A linear time complexity algorithm for creating a Huffman tree is shown in Algorithm 4.

B. Examples of Output of Discop

Table IV gives an example of employing random sampling and Discop with GPT-2 [47] to continue writing 200 tokens under $p = 0.95$ truncation. As can be seen, the semantics of texts are both relevant to the given context.

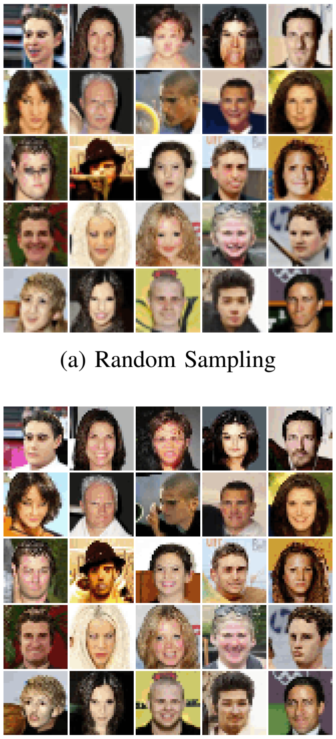
Fig. 7 gives examples of employing Image GPT [57] to perform image completion under $p = 0.95$ truncation. The upper half of each of them is obtained by down-sampling and

Algorithm 4: create_tree$(\mathcal{P}^{(t)})$: create a Huffman tree according to $\mathcal{P}^{(t)}$ in $O(|V|)$ time

```
Input: Distribution $\mathcal{P}^{(t)}$
Output: Root Node of the Created Huffman Tree root
indices, probs $\leftarrow \mathcal{P}^{(t)}$
nodes $\leftarrow$ [ ]
$n \leftarrow$ probs.size()
for $i \leftarrow 0$ to $n-1$ do
    node $\leftarrow$ Node(indices[$i$], probs[$i$], $\varnothing$, $\varnothing$)
    nodes.append(node)
$q_1 \leftarrow$ queue(nodes)
$q_2 \leftarrow$ queue()
subroutine GetMin()
    nonlocal $q_1$, $q_2$
    if $q_1$.size() $> 0$ and $q_2$.size() $> 0$ and
      $q_1$.front().prob $<$ $q_2$.front().prob then
        item $\leftarrow q_1$.front()
        $q_1$.pop_front()
    else if $q_1$.size() $= 0$ then
        item $\leftarrow q_2$.front()
        $q_2$.pop_front()
    else if $q_2$.size() $= 0$ then
        item $\leftarrow q_1$.front()
        $q_1$.pop_front()
    else
        item $\leftarrow q_2$.front()
        $q_2$.pop_front()
    return item
while $q_1$.size() + $q_2$.size() $> 1$ do
    left $\leftarrow$ GetMin()
    right $\leftarrow$ GetMin()
    prob $\leftarrow$ left .prob + right .prob
    $q_2$.push_back(Node($\varnothing$, prob, left, right))
if $q_2$.size() $> 0$ then
    root $\leftarrow q_2$.front()
else
    root $\leftarrow q_1$.front()
return root
```

cropping selected samples in the CelebA dataset, and we use random sampling and Discop to complete it. It can be seen that the quality of the image generated by random sampling is similar to that of the image generated by Discop. In order to generate a large number of images quickly, we choose the Image GPT model with the generated image size of 32×32 as an instance. If a larger model is chosen, or the ability of generative models becomes more powerful in the future, the quality of the image will be improved.

For the text-to-speech task, we have put some audio clips here[6]. Both the audio clips generated by random sampling and Discop sound clear and normal.


(a) Random Sampling

(b) Discop

Fig. 7: Samples of employing Image GPT to perform image completion under $p = 0.95$ truncation.

C. Steganalysis Experiments

Although we have proved the security of the steganography method based on "distribution copies" in Sec. III-A, we still try to distinguish between covers (generated by random sampling) and stegos (generated by steganographic sampling) with steganalyzers for the sake of the integrity of this work. Existing steganalysis methods generally 1) generate many covers and stegos under the same settings as the communication parties and then train a binary classifier; 2) use the trained classifier to predict whether a sample is a stego or a cover. Notably, the generated covers and stegos are in pairs, with the same contexts (for the text generation and image completion tasks) or with the same texts (for the text-to-speech task).

In our steganalysis experiments, we generate more samples to better train steganalyzers, and all covers and stegos are generated with the truncation parameter of $p = 0.95$. Except for the number of generated samples and the truncation parameter p, the settings are the same as described in Sec. V-A.

a) Text generation: We generate 10,000 covers and 10,000 stegos. We employ three linguistic steganalyzers, including FCN [65], R-BiLSTM-C [66], and BiLSTM-Dense [67].

b) Image Completion: We generate 10,000 covers and 10,000 stegos. We employ a widely used image steganalyzer: Steganalysis Residual Network (SRNet) [20].

c) Text-to-speech: We generate 1000 covers and 1000 stegos. We employ an audio steganalyzer: Combined Time and

[6] https://github.com/comydream/Discop/tree/main/audio_samples

TABLE IV: An example of employing GPT-2 to perform text generation under $p = 0.95$ truncation.

Context: I remember this film, it was the first film I had watched at the cinema.
Random Sampling: There was a last minute switch because I had watched it a decade ago in the City School balcony of a big theatre where the writers were also prominent. So I just watched it, cut it, then cut it. When I was about 15 years old, I went to the Millie Murphy director's ticket and watched The Girl who Would Get Away With Murder and then I watched an Ode to love. It's just magic. Once we got into that period with Baby Driver and Gravity and Rushmore, the characters played what I'd always dreamed. You know, they'd come up with creative, futuristic, ridiculously stupid songs from here and then the originals would come out, the normal birds would just sit there and wait for a substitution in the end credits to get on the schedule. And I just loved that whole process of watching this wonderful, there is so much greater potential here and it's been so exciting to continue to put in so much work and still succeed with these great movies in the works
Discop: I was very nervous, at the time. I was 22, and the director was actually at a birthday party in Paris and just saying the doors were closed. But I put on a large black motorcycle, got my album saved and drove on, doing videos about history and the films. So I was really excited. And I found that in this movie, we were at a concert in London. And that concert was where every song and every act was at the movie playing the music that could be heard and the message was, "By God, there's people. But time has passed, and we're here, maybe from another world. Let's see how things can go." And at the end of the movie I said, "I hope the thing happened a bit, because there's no way you can change what you know. You know? Something has happened. Just let it happen. Let it happen." And at the end, I say to myself, "Hey, maybe we can change

Frequency (CTaF) [68].

The detection error rate P_{E} of the testing set is used to evaluate the security of steganographic methods. For all steganalysis experiments, the generated datasets are divided into the training set, validation set, and testing set (3:1:1).

The experimental results are shown in Table V. The detection error rates against Discop deployed on three tasks are all close to 50%. The results show that these classifiers are challenging to distinguish between the covers and the stegos, which validates the security of Discop.

TABLE V: Steganalysis results for Discop.

Task	Steganalyzer	P_{E}
Text Generation	FCN [65]	50.10%
	R-BiLSTM-C [66]	50.45%
	BiLSTM-Dense [67]	49.95%
Image Completion	SRNet [20]	50.05%
Text-to-Speech	CTaF [68]	49.50%

D. Generation Quality Evaluation Experiments

Discop makes the stego distribution strictly equal to the cover distribution. It means that the quality of the steganographic generations is similar to that of normal generations. To verify this, we conduct quality evaluation experiments on the covers and stegos.

1) Text generation: We denote the cover and stego text datasets generated in Appendix C as T_{c} and T_{s}. For each dataset, we can obtain the mean and standard deviation of the perplexity (ppl) values of the texts, as shown in Table VI. It can be seen that they are close, indicating that the quality of the texts generated by steganographic sampling is similar to that generated by random sampling.

2) Image completion: We denote the cover and stego image datasets generated in Appendix C as $I_{\mathrm{c}0}$ and I_{s}, and additionally generated 10,000 covers with the same contexts as dataset $I_{\mathrm{c}1}$. We use the Fréchet Inception Distance (FID) [69] to measure the distance between $I_{\mathrm{c}0}$ and I_{s} (a cover dataset and a stego dataset) and the distance between $I_{\mathrm{c}0}$ and $I_{\mathrm{c}1}$ (two cover datasets, for reference), as shown in Table VII. It can be seen that they are close, indicating that the quality of the images generated by steganographic sampling is similar to that generated by random sampling.

3) Text-to-speech: We denote the cover and stego speech datasets generated in Appendix C as $A_{\mathrm{c}0}$ and A_{s}, and additionally generated 1000 covers with the same texts as dataset $A_{\mathrm{c}1}$. We employ the Perceptual Evaluation of Speech Quality (PESQ) [70], which measures the similarity between two audio clips. For each pair of audio clips with the same text from two datasets, we can calculate the PESQ between them. The mean and standard deviation of the PESQ values between $A_{\mathrm{c}0}$ and A_{s} (a cover dataset and a stego dataset) and that between $A_{\mathrm{c}0}$ and A_{c_1} (two cover datasets, for reference) are presented in Table VIII. It can be seen that they are close, meaning that the quality of the speeches generated by steganographic sampling is similar to that generated by random sampling.

TABLE VI: Quality evaluation of generated texts.

Dataset	mean(ppl)	std(ppl)
T_{c}	42.858	21.512
T_{s}	42.229	22.770

E. Toy Examples of Steganographic Distortions

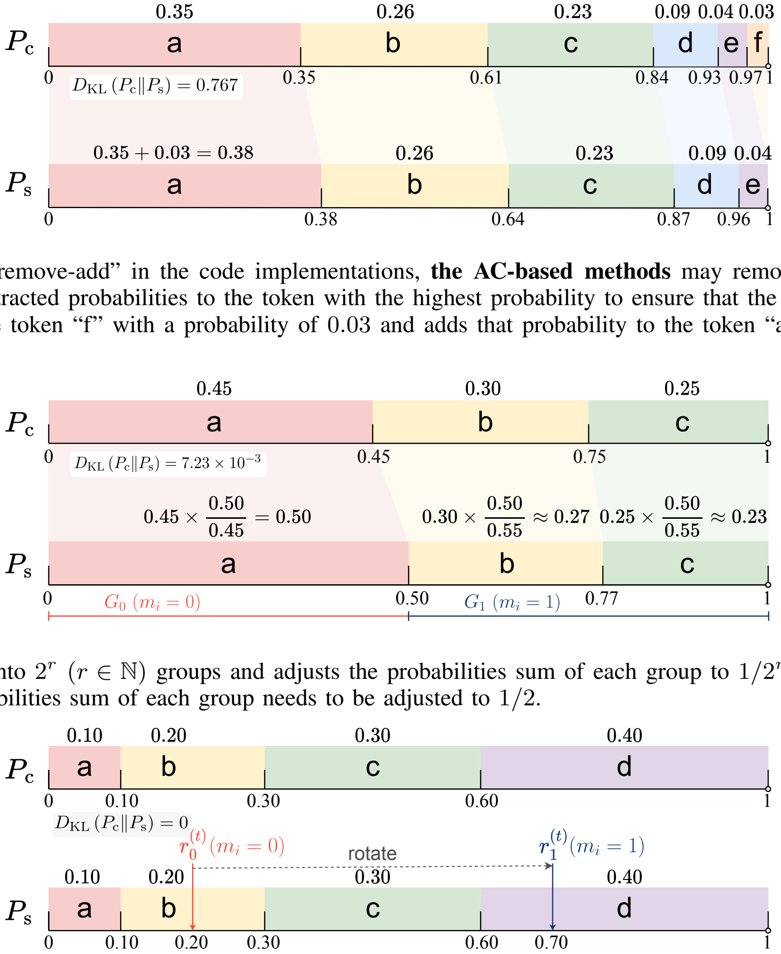
We provide toy examples in Fig. 8 to illustrate the distortions introduced by the AC-based methods and ADG, as well as how Discop can preserve the distribution.

TABLE VII: Quality evaluation of generated images.

Datasets		FID
I_{c0}	I_{s}	3.858
I_{c0}	I_{c1}	3.843

TABLE VIII: Quality evaluation of generated audios.

Datasets		mean(PESQ)	std(PESQ)
A_{c0}	A_{s}	3.354	0.122
A_{c0}	A_{c1}	3.356	0.128



(a) By “cutoff-rescale-round-remove-add” in the code implementations, **the AC-based methods** may remove the tokens with the lowest probabilities and add the subtracted probabilities to the token with the highest probability to ensure that the probabilities sum to 1. In this example, Meteor removes the token “f” with a probability of 0.03 and adds that probability to the token “a”, whose probability becomes 0.38 as a result.

(b) **ADG** divides all tokens into 2^r $(r \in \mathbb{N})$ groups and adjusts the probabilities sum of each group to $1/2^r$. In this example, the number of groups is 2, and the probabilities sum of each group needs to be adjusted to $1/2$.

(c) **Discop** creates “distribution copies” by rotation and then performs a random sampling from one of these copies. Since the distribution of all “distribution copies” is identical, Discop does not modify the distribution.

Fig. 8: The distortions introduced by ADG and the AC-based methods and how Discop can maintain the distribution.