Improved Generative Steganography Based on Diffusion Model

Qing Zhou, Ping Wei, Zhenxing Qian, *Senior Member, IEEE*, Xinpeng Zhang, *Senior Member, IEEE*, and Sheng Li, *Member, IEEE*

***Abstract*— The rapid growth of generative models has led to a new direction in steganography called generative steganography (GS). It allows message-to-image generation without the need for a carrier image. Recently, generative steganography methods have been proposed using generative adversarial networks (GANs) and Flow models. On the one hand, methods that use GANs to generate stego images struggle to fully recover the hidden message because the networks are not reversible. On the other hand, methods based on Flow encounter a problem where the images they create might not look real, mainly because the network has limitations in being reversible. Diffusion models fulfill network reversibility while generating high-quality images. However, the framework of existing diffusion models is reversible, but hidden message recovery is not perfectly reversible, resulting in the recovered message being similar but not exactly the same as the hidden one. Existing diffusion models are typically trained for one-directional image generation tasks, so they face some problems when dealing with bi-directional steganography tasks. If pre-trained diffusion models are directly used to generate stego images, exact secret data extraction through the diffusion process cannot be achieved. In this paper, we present an improved generative steganography based on the diffusion model (GSD), which conceals secret data in the frequency domain of random noise to enhance the security and accuracy of steganography, and re-trains the denoising diffusion implicit model (DDIM) for steganography, called the StegoDiffusion. During training StegoDiffusion, random noise is injected into the clean natural images and then trained through the forward diffusion process to obtain the re-trained StegoDiffusion. Our proposed GSD scheme achieves a 100% extraction accuracy for hidden secret data with a payload of 1 bit-per-pixel (bpp) in a single channel, and generates high-quality stego images in PNG format.**

***Index Terms*— Generative steganography, diffusion model, DDIM.**

I. Introduction

STEGANOGRAPHY is the practice of concealing secret data within other non-secret data to hide the existence of the concealed data. This method works with different kinds of media, like images [1], [2], videos [3], [4], audio [5], and text [6], without changing the appearance of the original content. Among these, digital images are widely used as cover mediums, and the result with hidden data is referred to as the stego image. The objective is to ensure that hidden data remain unnoticed by unintended observers. Steganography is often used for secure communication, data protection, or covert operations and plays a critical role in digital security and privacy. Steganography aims to keep secret data safe during transmission, preventing unauthorized access. On the other side, steganalysis checks if an image, whether it is the original (cover) or altered (stego), carries hidden information [7], [8]. Due to the specific rules and patterns followed by traditional steganography, specialized steganalysis tools can often detect these hidden messages. The emergence of generative steganography increases the difficulty of detection because the generated data is more realistic, making traditional steganalysis methods less sensitive.

Generative steganography is an evolved form that utilizes generative models, often based on deep learning techniques, to embed secret data. The primary motivation behind the development of generative steganography is to enhance the security and stealthiness of steganographic techniques. Generative steganography uses models like GANs or Flow to understand how the original data look and creates new data that seems realistic. During this process, secret data are embedded into the generative model, making the generated image carry hidden information imperceptible to the naked eye. Liu et al. [9] introduced a method to hide secret information in GANs using class labels. They created different types of stego images to represent the secret data. Likewise, Liu et al. [10] and Hu et al. [11] linked secrets with different value ranges in GANs. They generated stego images using the image generator and trained an extractor to get back the hidden data. Yet, their effectiveness falls short when it comes to payload, secret data extraction accuracy, and image quality, especially compared to conventional modification-based steganography methods. Later, Wei et al. [12] suggested a method to hide more secret data in the cover image. They created a generator that adds secret data while generating images, resulting in better stego images with more hidden data. This approach trained a dedicated extractor to recover hidden data. However, when the stego image payload exceeds 0.33 bit per pixel (bpp), the extraction accuracy drops significantly. To improve the accuracy of recovering secret data, Wei et al. [13] utilized the reversible Flow model to serve as both a generator and an extractor, establishing a bidirectional mapping between secret data and stego images. However, the image quality generated

Received 25 April 2024; revised 14 August 2024; accepted 2 February 2025. Date of publication 7 February 2025; date of current version 4 July 2025. This work was supported by the National Natural Science Foundation of China under Grant U22B2047 and Grant 62450067. This article was recommended by Associate Editor T. Zhang. *(Corresponding authors: Zhenxing Qian; Ping Wei.)*

Qing Zhou, Zhenxing Qian, Xinpeng Zhang, and Sheng Li are with the School of Computer Science, Fudan University, Shanghai 200438, China (e-mail: qingzhou21@m.fudan.edu.cn; zxqian@fudan.edu.cn; zhangxinpeng@fudan.edu.cn; lisheng@fudan.edu.cn).

Ping Wei is with the National Pilot School of Software, Yunnan University, Kunming, Yunnan 650091, China (e-mail: weip@ynu.edu.cn).

Digital Object Identifier 10.1109/TCSVT.2025.3539832

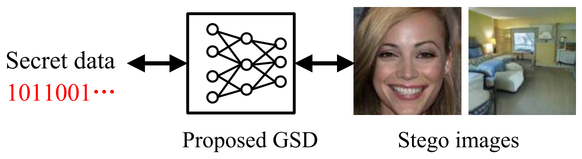


Fig. 1. The reversibility of Generative Steganography. Our proposed GSD scheme can directly generate stego images from secret data and then exactly recover the hidden secret data from stego images.

in [13] is unsatisfactory. This is due to the limitations imposed by the requirement of invertibility of Flow models. Compared to Flow models, diffusion models [14], [15] break free of the constraint of requiring a reversible network structure and achieving network invertibility. Using diffusion models for generative steganography may lead to breakthroughs.

However, three challenges arise when attempting to implement generative steganography using the diffusion model. (1) Diffusion models are unable to produce high-quality images when Gaussian noise is substituted with secret data in the spatial domain, causing a disturbance in the Gaussian noise distribution. (2) The accumulation of minor errors in the spatial domain during the forward and backward processes of the diffusion model results in a reduction in the extraction accuracy of secret data. (3) To preserve the input distribution of the generative models, some generative steganography methods [16], [17] utilize reject sampling to map secret data into Gaussian distribution, and then generate stego images using pre-trained generative models. However, the focus of provably secure steganography methods is to ensure security, without considering the practicality of the approach. The stego images generated by these methods are in floating-point format and are not saved as PNG or other storage formats. This makes these steganography methods impractical for real-world applications. Addressing the aforementioned challenges, we try to hide secret data within the frequency-domain coefficients of Gaussian noise. Then, we try using a pre-trained diffusion model [15], [18] to generate high-quality stego images through the denoising generative process. The receiver needs to extract the hidden data through the noise-adding diffusion process. However, the pre-trained diffusion model is not designed to process stego images with random noise (which can represent hidden data), leading to an inability to achieve an extraction accuracy 100%. To further enhance the accurate extraction of secret data, we try to retrain the denoising diffusion implicit model (DDIM) [15] for handling stego images.

In this paper, we propose an innovative GSD generative steganography scheme based on DDIM [15], which hides secret data in the frequency domain of random latents obeying the Gaussian distribution, improving the security, stability, and accuracy of steganography. Moreover, the GSD scheme retrains a reversible diffusion model called StegoDiffusion. This enhances the creation of high-quality stego images and ensures the precise extraction of secret data. As seen in Fig. 1, the GSD method changes binary secret information into stego images in PNG format and can accurately retrieve hidden data. Due to the reversible nature of the diffusion model, GSD can create and retrieve messages using just one diffusion model. This simplifies secure communication between the sender and receiver by exclusively sharing the StegoDiffusion model and the concealment method. The key contributions of our GSD scheme are as follows:

- To our knowledge, we are the first to investigate the task of **g**enerative **s**teganography based on **D**DIM (GSD).
- We propose hiding secret messages in the frequency domain of Gaussian noise, thus avoiding the impact of accumulated errors in the time domain on the hidden secret data.
- We present a diffusion model designed specifically for steganography, known as StegoDiffusion. It enables a bidirectional connection between stego images and stego latents.
- In practical use, GSD shows superiority over existing methods, especially in achieving higher extraction accuracy with the same payloads.

II. Preliminaries

A. Modification-Based Steganography

Conventional image steganography involves hiding secret data by modifying the cover image. The type of data embedding can be classified into spatial domain embedding and frequency domain embedding algorithms, depending on the data domain. Spatial domain embedding steganography was the first to be developed, with its fundamental concept involving the direct alteration of the pixel values in the carrier image to hide secret message. Typical spatial domain embedding steganographic algorithms include the Least Significant Bit (LSB) steganography algorithm [19], the Least Significant Bit Match (LSBM) steganography algorithm [20], and the random modulation steganography algorithm [21], [22]. Frequency domain embedding steganographic algorithms often utilize JPEG images as carriers. JPEG compression employs the Discrete Cosine Transform (DCT) technique to embed secret message within the DCT coefficients. The earliest algorithm, JSteg, replaces the least significant bit in DCT coefficients [23]. F5 [24] aims for uniform distribution and nsF5 [25] counters histogram-based detection. Outguess employs a two-round strategy to align DCT coefficient histograms between cover and stego images [26]. However, modifying the cover image inevitably leads to distortion, which can be easily detected by steganalysis.

To reduce distortion and enhance security, Filler et al. first proposed an adaptive steganography method to minimize distortion based on syndrome-trellis codes (STC) [27], manually designing an embedding distortion function to minimize overall distortion. Later, various adaptive steganographic methods with distinct distortion functions were proposed [28], [29], [30], [31]. However, the manual design approach makes it challenging to minimize image distortion. With the advancement of deep learning, several adaptive steganographic methods have emerged that automatically learn the embedding losses through deep neural networks [32]. Tang et al. suggested a way to automatically learn how to hide information using a generative adversarial network (GAN) [33]. By alternating training of a generator and a discriminator, the framework automatically learns the probabilities of pixel-level distortion. Later, an end-to-end trainable framework HiDDeN [34] was proposed to achieve robust steganography using imperceptible perturbations. Hayes et al. introduced a robust steganographic algorithm via adversarial training [35]. StegaStamp [36] used deep neural networks to learn robust encoding/decoding algorithms, allowing the concealment of data within digital images. Even after operations like printing and re-capturing, the hidden data can still be extracted from the distorted images. Similarly, additional adaptive steganography schemes were suggested utilizing GAN [37], [38], [39], [40]. Recently, Yao et al. focused on robust adaptive steganography to withstand lossy attacks [41]. This is achieved by minimizing the number of check codes to improve security without sacrificing robustness. Moreover, researchers have explored methods for achieving high-capacity steganography by concealing secret images within cover images using Invertible Neural Networks (INN) or Auto-Encoder. Jing et al. [42] and Lu et al. [43] introduced two image hiding schemes based on reversible networks, which conceal secret images within another image. Baluja et al. introduced a method to hide color images in another with minimal loss of quality by the Auto-Encoder [44].

All the mentioned methods above are modification-based steganography. However, modifying the cover image inevitably introduces image distortions, which are the clues that steganalysis tools aim to detect. Facing this challenge, researchers have begun to explore novel steganographic approaches, with generative steganography emerging as a notable direction.

B. Generative Steganography

Generative steganography uses models like Generative Adversarial Networks (GANs) or Flow models to learn the distribution of original data and create new data that looks convincingly real. This approach involves embedding secret information within the generative model, allowing the generated data to carry hidden information in a manner that is imperceptible in appearance. The existing generative steganographic techniques can be categorized into two groups: those based on GANs and those based on Flows.

Researchers explore the use of generative adversarial networks (GANs) in steganography by converting secret messages into intermediate data like image labels or semantic information. Liu et al. [9] translated messages into class labels using ACGAN [45], while Zhang et al. [46] mapped messages to semantic information for GAN-based image generation. To overcome the low capacity of simple labels, some researchers encode messages as noise signals. For instance, Hu et al. [11], [47] used DCGANs [48], Li et al. [49] employed Wasserstein GAN Gradient Penalty (WGAN-GP) [50] generative steganography. Nevertheless, effectively retrieving data from the generated images, particularly with increased hiding capacities, continues to be a challenge because of the unidirectional nature of GANs.

In addition, researchers also pay attention to Flow-based generative steganography. After mapping secret data to Gaussian noise using a specially designed mapping method, Zhou et al. [51] directly used the pre-trained Glow model to generate stego images. To improve the precision of extracting secret data, Wei et al. [13] used the Glow [52] model to serve as both generator and extractor. This allowed forward and backward mapping between secret data and stego images. However, the stego images created in this study showed slightly lower quality compared to those produced by GANs. The quality difference occurs because Flow models require invertibility, imposing limitations on overall quality.

In summary, current GAN-based generative steganography methods face challenges in fully retrieving concealed secret data because of the absence of network invertibility. Meanwhile, Flow-based methods generate unrealistic images, attributing to the inherent reversibility restriction of the generative model. Recently, diffusion models have demonstrated excellent generative capabilities and the diffusion model network structure is reversible. Recent work [53] achieved provably secure steganography based on pre-trained denoising diffusion probabilistic models (DDPMs), mapping secret data into the difference between intermediate and final generated images. It focused on security neglecting extraction accuracy in practical applications. If the generated images are stored in PNG format, the extraction accuracy will quickly decrease. In this paper, to achieve both high extraction accuracy and high image quality, we try merging the diffusion model into generative steganography. The main challenge lies in the fact that pre-trained diffusion models are specifically trained for a unidirectional task, generating stego images. However, they struggle with bi-directional tasks, particularly in accurately extracting hidden secrets from generated stego images.

C. The Denoising Diffusion Implicit Model

Currently, the diffusion model is one of the most popular ways to create images [54], [55]. It beats GAN in image quality and shows amazing visual effects [18], [56], [57], [58], [59]. The diffusion model has two steps: forward diffusion and backward image generation. During the diffusion phase, noise is introduced incrementally to the image, resulting in a noisy image that adheres to a Gaussian distribution. Subsequently, in the generation stage, the noisy image, characterized by a Gaussian distribution, is progressively cleaned up to yield a clean image. Later, the Denoising Diffusion Probabilistic Model (DDPM) [14] used the diffusion model to create clear images in high detail, setting a new direction for image generation. Later, OpenAI enhanced DDPM by incorporating a classification-guided approach for image generation, surpassing the image quality produced by Generative Adversarial Networks (GANs) [56]. OpenAI also released the GLIDE model [57], which achieves text-to-image generation by incorporating the CLIP model [60] or through classifier-free guidance. Subsequently, OpenAI introduced a two-stage generative model called DALLE-2 [58], which first converts the input prompt into the corresponding image embeddings using CLIP and then generates the final image through a decoder. In the same year, Google presented the text-to-image model Imagen [59], which transforms text input into text embeddings and subsequently creates images through diffusion models. Latent Diffusion Models (LDMs) [18] operate in a latent space, enabling denoising and generation tasks, and they enhance capabilities by considering multiple conditions during the generation process. Then, StableDiffusion, a version of LDMs, ensures the stability and reliability of data, thereby enhancing the efficiency and effectiveness of generative models. However, the process of generating images in DDPM involves numerous steps and takes a considerable amount of time. In order to improve the effectiveness of sampling, Song et al. [15] proposed the denoising diffusion implicit model (DDIM). DDIM employs a non-Markov chain approach to minimize the required sampling iterations.

In DDPM, the forward diffusion process is, in fact, based on the marginal distribution $q(\mathbf{x}_t \mid \mathbf{x}_0)$ rather than using the transition function $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$. In other words, the forward diffusion process only needs to satisfy the marginal distribution. Hence, DDIM improves the diffusion process into a non-Markov chain process using marginal distributions. Throughout this procedure, the creation of $\mathbf{x}_t$ is not solely connected to the preceding $\mathbf{x}_{t-1}$ but is also impacted by the initial input $\mathbf{x}_0$. The specific probability distribution of this diffusion process is presented below.

$$q_\sigma(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = q_\sigma(\mathbf{x}_T \mid \mathbf{x}_0) \prod_{t=2}^{T} q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0), \tag{1}$$

in the above equation, $\mathbf{x}_0$ represents the initial input, which is a clean image, while $\mathbf{x}_t$ corresponds to the image with noise at step t, and q_σ represents the transition probability with respect to σ during the diffusion process. The parameter σ regulates the level of randomness in the diffusion process. Specifically, $q_\sigma(\mathbf{x}_T \mid \mathbf{x}_0) = N\left(\mathbf{x}_T; \sqrt{\bar{\alpha}_T}\mathbf{x}_0, (1-\bar{\alpha}_T)\mathbf{I}\right)$, where $\mathbf{I}$ is the identity matrix. With $t > 1$, we have:

$$q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = N\left(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I}\right), \tag{2}$$

at step t, $\bar{\alpha}_t$ represents a hyperparameter.

The derivation of the forward diffusion process follows Bayes' theorem:

$$q_\sigma(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)\, q_\sigma(\mathbf{x}_t \mid \mathbf{x}_0)}{q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_0)}. \tag{3}$$

In non-Markovian diffusion processes, the generation of $\mathbf{x}_t$ is influenced by both $\mathbf{x}_0$ and $\mathbf{x}_{t-1}$ simultaneously.

During the training phase, when provided with an input image $\mathbf{x}_0$ and randomly chosen noise $\epsilon_t \sim N(0, 1)$, $\mathbf{x}_t$ is derived through the marginal probability in the forward diffusion process:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t \tag{4}$$

After training, the trained neural network can provide a prediction for the noise value ϵ_t as $\epsilon_\theta(\mathbf{x}_t, t)$. Consequently, a predicted value for the original clean image $\mathbf{x}_0$ can be obtained:

$$\mathbf{x}_0' = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}. \tag{5}$$

Replacing the predicted value $\mathbf{x}_0'$ into (2), we get:

$$\begin{aligned} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) &\approx q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0 = \mathbf{x}_0') \\ &= N(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}) \\ &\quad + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\epsilon_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}). \end{aligned} \tag{6}$$

Expanding the above formula, the formula for sampling $\mathbf{x}_{t-1}$ from $\mathbf{x}_t$ can be derived:

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}\right)}_{\mathbf{x}_0'} + \underbrace{\sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\epsilon_\theta(\mathbf{x}_t, t)}_{\text{pointing to } \mathbf{x}_t} + \underbrace{\sigma_t\epsilon_t}_{\text{random noise}}, \tag{7}$$

in this case, σ_t can be defined as η times the square root of $(1-\bar{\alpha}_{t-1})/(1-\bar{\alpha}_t)$ multiplied by the square root of $1-\bar{\alpha}_t/\bar{\alpha}_{t-1}$, where η is within the range [0, 1]. The variable ϵ_t follows a normal distribution with mean 0 and variance 1, and $\epsilon_\theta(\mathbf{x}_t, t)$ denotes the estimated value of ϵ_t.

As shown in Eq. 6, in DDPM, the transition probability $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ in each step of the backward generation process approximates the posterior probability $q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ in the forward diffusion process. Similarly, DDIM shares the same training process and objective as DDPM. Both methods aim to maximize the likelihood distribution of the image $\mathbf{x}_0$, and both use variational lower bounds to maximize the likelihood function.

$$\begin{aligned} -\mathbb{E}_{\mathbf{x}_0}\log p_\theta(\mathbf{x}_0) &= \mathbb{E}_{q_\sigma(\mathbf{x}_{0:T})}[\log q_\sigma(\mathbf{x}_T \mid \mathbf{x}_0) \\ &\quad + \sum_{t=2}^{T}\log q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \\ &\quad - \sum_{t=1}^{T}\log p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) - \log p_\theta(\mathbf{x}_T)] \\ &= J_\sigma, \end{aligned} \tag{8}$$

here, J_σ represents a variational lower bound on the maximum likelihood function. As stated in the original paper [15], the variational lower bound of DDIM is related to the loss function of DDPM:

$$J_\sigma = \gamma \cdot L_d + C, \tag{9}$$

where $\gamma \in R^+$ and $C \in R$, L_d is the loss function of DDPM. Therefore, the optimization goal for the variational lower bound J_σ in DDIM can be aligned with that of DDPM, making their training processes identical. DDIM can undergo training either through the network training method applied in DDPM or by employing a pretrained DDPM model directly for image generation.

In this paper, We utilize DDIM for the steganography task. An ordinary differential equation (ODE) is constructed, and the approximate Eulerian iterative formula is employed to generate stego images and retrieve hidden data. This procedure depends on utilizing a pre-existing one-directional generative network to achieve perfect reversibility of data in the bi-directional steganographic task.

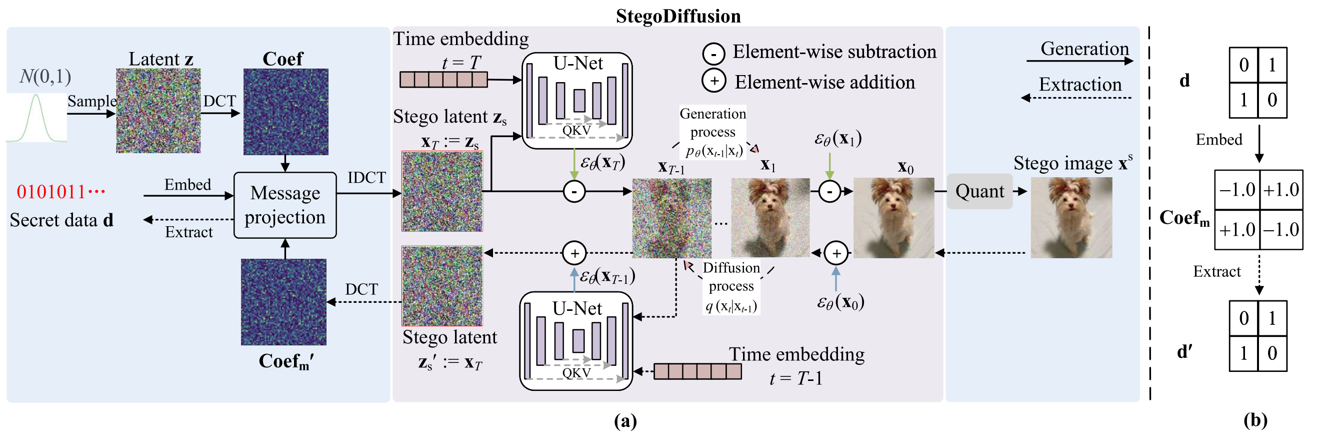


Fig. 2. An Overview of the Proposed Generative Steganography Diffusion (GSD) Scheme. (a) The generation and extraction process of GSD. (b) The illustration of the message projection process when secret data are '0110'.

III. Proposed Scheme

This paper introduces a new steganography method called GSD, built upon DDIM, to achieve Gaussian distribution-preserving data hiding, high-quality stego image generation in PNG format, and exact secret data extraction. The proposed GSD, as shown in Fig. 2, consists primarily of two crucial components: the embedding/recovering of secret data and the StegoDiffusion model. During stego image generation, the sender randomly samples a latent variable $\mathbf{z}$ from Gaussian distribution, $\mathbf{z} \sim N(0, 1)$, and transforms the latent $\mathbf{z}$ into the frequency domain by discrete cosine transformation (DCT). Next, we hide secret data $\mathbf{d}$ within the frequency coefficients of random latent $\mathbf{z}$ by replacing the coefficients with binary secret data. Then, convert the modified coefficients back to the spatial domain using inverse DCT (I-DCT) to get the stego latent $\mathbf{z}_s$. The $\mathbf{z}_s$ is employed as the starting noisy image $\mathbf{x}_T$, which then undergoes the StegoDiffusion generative process to produce a stego image $\mathbf{x}_0$, $\mathbf{x}_T \to \mathbf{x}_{T-1} \cdots \to \mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0$. Following quantization, a last stego image $\mathbf{x}^s$ can be produced and saved in hardware as PNG files. Then, the sender can transmit the stego image $\mathbf{x}^s$ to the receiver using lossless transmission through social media platforms, such as the original image transmission mode in WeChat. Upon receiving the stego PNG image, the receiver downloads the PNG image in the hardware and then reads from the hardware to extract hidden data through the shared StegoDiffusion. Once the stego image is read from the hardware, the data $\mathbf{x}^s$ is initially converted into the floating-point format and input into StegoDiffusion. This process progressively reconstructs the original noisy image through the diffusion steps, $\mathbf{x}^s \to \mathbf{x}_0 \to \mathbf{x}_1 \cdots \to \mathbf{x}_{T-1} \to \mathbf{x}_T$. Subsequently, $\mathbf{x}_T$ serves as the recovered stego latent representation $\mathbf{z}'_s$, from which the hidden data $\mathbf{d}'$ is extracted from the frequency coefficients of $\mathbf{z}'_s$.

A. Data Hiding

In this part, we explain how to hide secret data in the frequency domain of latent $\mathbf{z}$, and then demonstrate that our embedding method can generate a distribution similar to the Gaussian distribution.

To address the three challenges, we hide secret data in the frequency domain of latent $\mathbf{z}$: (1) Concealing data in the spatial domain of latent $\mathbf{z}$ through replacement might alter the mean and standard deviation of the Gaussian distribution. (2) The secret data concealed in the spatial domain of latent $\mathbf{z}$ could be influenced by the error accumulation of the diffusion model. (3) In real-world applications, the use of quantized stego images for storage or transmission often results in distortion of the secret data.

First, randomly sample a latent $\mathbf{z}$ from standard Gaussian distribution, $\mathbf{z} \sim N(0, 1)$, and then transform it to the frequency domain by DCT,

$$\mathbf{Coef} = \mathrm{DCT}(\mathbf{z}). \tag{10}$$

then, modify each coefficient $Coef$ according to each binary secret data d to obtain $Coef_m$:

$$\begin{aligned} Coef_m &:= (Coef \leftarrow +1.0, \quad if \quad d = 1) \\ Coef_m &:= (Coef \leftarrow -1.0, \quad if \quad d = 0) \end{aligned} \tag{11}$$

When the sender wants to conceal the secret data "1," the related DCT coefficient is adjusted to "+1.0"; when hiding the secret data "0," the corresponding DCT coefficient is adjusted to "−1.0". From the given equations, it is clear that the adjusted DCT coefficients $\mathbf{Coef}_m$ are entirely based on the secret data input $\mathbf{d}$. This allows us to simplify the above representation as follows:

$$\mathbf{Coef}_m = 1.0 \odot \mathrm{reshape}(2 \times \mathbf{d} - 1), \tag{12}$$

where $\mathbf{d}$ is the sequence of secret data; $\mathbf{Coef}_m$ denotes the DCT coefficients embedded with secret data; $\odot$ denotes element-wise multiplication. The function reshape$(\cdot)$ transforms the input into a matrix that matches the shape of the DCT coefficients of $\mathbf{z}$.

After embedding secret data in the DCT coefficients, the stego latent $\mathbf{z}_s$ can be obtained by IDCT transformation,

$$\mathbf{z}_s = \mathrm{IDCT}(\mathbf{Coef}_m). \tag{13}$$

We use the Monte Carlo simulation technique [61] to analyze the distribution of latent stego $\mathbf{z}_s$. At first, we randomly sample 10 latent variables $\mathbf{z}$ from a standard Gaussian

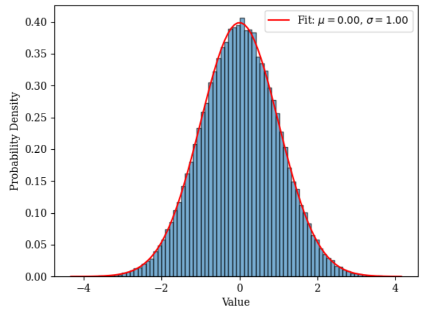


Fig. 3. Histogram and fitted distribution of stego latent $\mathbf{z}_s$.

Algorithm 1 Training Strategy of StegoDiffusion

```
Data: Number of sampling steps $\phi$; hyper-parameters $\bar{\alpha}_t$
    for each step ($t \in [1, \phi]$); initialed network U-net.
Result: The trained StegoDiffusion model.
1 while in sampling steps do
2     Randomly sample an input image $\mathbf{x}_0$ from dataset;
3     Inject random noise into $\mathbf{x}_0$ by updating it:
      $\mathbf{x}_0 := \mathbf{x}_0 + \mathbf{n}$, $\mathbf{n} \sim N(0, 0.01^2)$;
4     Select a diffusion step $t$ randomly from a uniform
      distribution $U(1, T)$;
5     Select added noise $\epsilon$ randomly from a standard
      Gaussian distribution $N(0, 1)$.;
6     Calculate the loss $L_d$ and adjust U-net using
      gradients: $\nabla_\theta \left\| \epsilon - \epsilon_\theta \left( \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t \right) \right\|^2$.
```

distribution, where $\mathbf{z} \sim N(0, 1)$ and each $\mathbf{z}$ with a size of 64×64 and 3 channels. Using our hiding method, we generate the corresponding stego latents $\mathbf{z}_s$ for these 10 latents $\mathbf{z}$, resulting in a total of 30 single channel $\mathbf{z}_s$ latents and involving 122880 statistical values. Then we compute the histogram of these 122880 data points and fit its distribution. The fitting result indicates a Gaussian distribution with a mean of 0 and a standard deviation of 1, shown in Fig. 3. This shows that the stego latent $\mathbf{z}_s$ follows a Gaussian distribution, and the distribution of $\mathbf{z}_s$ is very close to that of latent $\mathbf{z}$.

B. StegoDiffusion

In our scheme, StegoDiffusion is a modified version of DDIM designed specifically for handling stego images. It follows a similar training strategy to DDIM, with the main distinction being the addition of Gaussian noise to $\mathbf{x}_0$ while pre-processing to guarantee precise data retrieval. The algorithm 1 outlines the StegoDiffusion training process. Hyperparameters $\bar{\alpha}_t$ should be set in advance. In each iteration, a training image is randomly selected and random noise is injected to serve as $\mathbf{x}_0$. Then, a diffusion step t is randomly selected, along with additional random noise ϵ. The U-net neural network generates $\epsilon_\theta(\mathbf{x}_t, t)$ as its estimate of the added noise ϵ at that specific step. Finally, the parameters of the U-net are adjusted based on the distance between the predicted noise and the actual noise, with the goal of making the predicted noise $\epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)$ closer to the actual added noise ϵ.

In StegoDiffusion, a sender can generate stego images through the generation process, and then a receiver can accurately extract data from these stego images through the diffusion process, enabling data reversibility rather than just framework reversibility. In the generation process, as shown in Eq. 7, the coefficient of the random noise term is denoted by σ_t when predicting $\mathbf{x}_{t-1}$ from $\mathbf{x}_t$. When $\eta = 0$, the coefficient σ_t of random noise term is 0, resulting the overall random noise term being zero. Consequently, the generation of stego images becomes a deterministic process. Rearranging Eq. 7, we get the deterministic generation as:

$$\mathbf{x}_{t-1} = \frac{\mathbf{x}_t}{\sqrt{\alpha_t}} + \left(\sqrt{1-\bar{\alpha}_{t-1}} - \sqrt{\frac{1-\bar{\alpha}_t}{\alpha_t}} \right) \epsilon_\theta(\mathbf{x}_t, t). \tag{14}$$

in the genration process, if the initial noise image $\mathbf{x}_T$ follows a standard Gaussian distribution $N(0, 1)$ and undergoes T steps of denoising iterations according to the above formula, a content-determined image $\mathbf{x}_0$ can be generated.

Rearranging the Eq. 14 mentioned above, we can derive the transition probability $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ for the process of generating images as shown below.

$$\frac{\mathbf{x}_{t-1}}{\sqrt{\bar{\alpha}_{t-1}}} = \frac{\mathbf{x}_t}{\sqrt{\bar{\alpha}_t}} + \left(\sqrt{\frac{1-\bar{\alpha}_{t-1}}{\bar{\alpha}_{t-1}}} - \sqrt{\frac{1-\bar{\alpha}_t}{\bar{\alpha}_t}} \right) \epsilon_\theta\left(\mathbf{x}_t, t\right), \tag{15}$$

let $\Delta t = 1$, the Eq. 15 can be transformed as:

$$\frac{\mathbf{x}_{t-\Delta t}}{\sqrt{\bar{\alpha}_{t-\Delta t}}} - \frac{\mathbf{x}_t}{\sqrt{\bar{\alpha}_t}} = \left(\sqrt{\frac{1-\bar{\alpha}_{t-\Delta t}}{\bar{\alpha}_{t-\Delta t}}} - \sqrt{\frac{1-\bar{\alpha}_t}{\bar{\alpha}_t}} \right) \epsilon_\theta\left(\mathbf{x}_t, t\right), \tag{16}$$

Assuming $\sigma = (\sqrt{1-\bar{\alpha}}/\sqrt{\bar{\alpha}})$ and $\bar{\mathbf{x}} = \mathbf{x}/\sqrt{\bar{\alpha}}$, substituting into Eq. 16 yields an ordinary differential equation (ODE):

$$\mathrm{d}\bar{\mathbf{x}}(t) = \epsilon_\theta\left(\frac{\bar{\mathbf{x}}(t)}{\sqrt{\sigma^2+1}}, t \right) \mathrm{d}\sigma(t). \tag{17}$$

Euler's method is one of the fundamental techniques for numerically solving ordinary differential equations (ODEs). It is an iterative method that approximates the solution of a differential equation through a series of small steps. In the above equation, any $\mathrm{d}\bar{\mathbf{x}}(t)$ can be estimated using the Euler iteration method. Subsequently, on the basis of $\bar{\mathbf{x}}_{t\pm1} = \bar{\mathbf{x}}_t \pm \mathrm{d}\bar{\mathbf{x}}(t)$, we can solve $\bar{\mathbf{x}}_{t\pm1}$. Given $\mathbf{x}_{t\pm1} = \sqrt{\bar{\alpha}}\bar{\mathbf{x}}_{t\pm1}$, we can then obtain $\mathbf{x}_{t\pm1}$. Thus, the image $\mathbf{x}_{t\pm1}$ corresponding to any step t can be solved by ODE, demonstrating that the StegoDiffusion model is reversible.

When $\Delta = -1$ in Eq. 16, the diffusion process's transition probability $q(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$ is obtained in the following manner:

$$\begin{aligned} &\frac{\mathbf{x}_{t+1}}{\sqrt{\bar{\alpha}_{t+1}}} - \frac{\mathbf{x}_t}{\sqrt{\bar{\alpha}_t}} \\ &\quad = \left(\sqrt{\frac{1-\bar{\alpha}_{t+1}}{\bar{\alpha}_{t+1}}} - \sqrt{\frac{1-\bar{\alpha}_t}{\bar{\alpha}_t}} \right) \epsilon_\theta\left(\mathbf{x}_t, t\right), \\ &\mathbf{x}_{t+1} = \sqrt{\alpha_{t+1}}\mathbf{x}_t + \left(\sqrt{1-\bar{\alpha}_{t+1}} - \sqrt{\alpha_{t+1} - \bar{\alpha}_{t+1}} \right) \epsilon_\theta\left(\mathbf{x}_t, t\right). \end{aligned} \tag{18}$$

In summary, the generation and diffusion processes of StegoDiffusion are entirely reversible. When the initial noise follows a Gaussian distribution $\mathbf{x}_T \sim N(0,1)$ and the number of iterations is T, denoising step by step throughout the generation process yields a clean image $\mathbf{x}_0$, as indicated in Eq. 15. Simultaneously, if the image $\mathbf{x}_0$ is used as input, adding noise step by step through the diffusion process allows for the gradual recovery of the initial noise $\mathbf{x}_T$, see Eq. 18. In StegoDiffusion, these two processes can achieve data reversibility, and the impact of the noise addition/removal process on the recovery of the original noise $\mathbf{x}_T$ can be nearly neglected.

C. Stego Image Generation

Throughout the diffusion model training, as the number of iteration steps increases in the noise addition process, the noised image $\mathbf{x}_t$ becomes more similar to a standard Gaussian distribution. When training StegoDiffusion, the upper limit for the iteration steps t (where t ranges from 1 to ϕ) is defined as $\phi = 1000$. However, following the DDPM generation process, the image generation process also requires denoising for the ϕ steps. This generation process is time-consuming and presents a disadvantage for covert communication.

Upon finishing the model training, in order to minimize the time needed for image generation in real-world scenarios, StegoDiffusion makes use of the fast generation technique introduced in [15] to produce stego images. As shown in Algorithm 1, during model training, it is necessary to go through the complete denoising process for ϕ steps. Only after completing the training, during image generation, can some intermediate steps in $[1,\phi]$ be skipped for accelerated generation. During accelerated generation, a subsequence $\tau = \{t_1, t_2, \cdots, t_{i-1}, t_i, \cdots, t_T\}$ can be sampled from the original steps $\{1,2,\cdots,\phi\}$, where each step t_i corresponds to the hyperparameter $\sqrt{\bar{\alpha}_{t_i}}$. The generation process can then be expressed as:

$$\mathbf{x}_{t_{i-1}} = \sqrt{\bar{\alpha}_{t_{i-1}}}\left(\frac{\mathbf{x}_{t_i} - \sqrt{1-\bar{\alpha}_{t_i}}\epsilon_\theta\left(\mathbf{x}_{t_i}, t_i\right)}{\sqrt{\bar{\alpha}_{t_i}}}\right) + \sqrt{1-\bar{\alpha}_{t_{i-1}} - \sigma_{t_i}^2}\cdot\epsilon_\theta\left(\mathbf{x}_{t_i}, t_i\right) + \sigma_{t_i}\epsilon, \tag{19}$$

in this case, the cofficient of random noise ϵ is given by $\sigma_{t_i} = \eta\sqrt{(1-\bar{\alpha}_{t_{i-1}})/(1-\bar{\alpha}_{t_i})}\sqrt{1-\bar{\alpha}_{t_i}/\bar{\alpha}_{t_{i-1}}}$, where the value of η varies between 0 and 1. When $\eta = 0$, the additional noise $\sigma_{t_i}\epsilon$ becomes zero, leading to a deterministic process for generating images. In this situation, transition probability $p_\theta(\mathbf{x}_{t_{i-1}} \mid \mathbf{x}_{t_i})$ in the fast and deterministic generation process is represented as:

$$\mathbf{x}_{t_{i-1}} = \sqrt{\bar{\alpha}_{t_{i-1}}}\left(\frac{\mathbf{x}_{t_i} - \sqrt{1-\bar{\alpha}_{t_i}}\epsilon_\theta\left(\mathbf{x}_{t_i}, t_i\right)}{\sqrt{\bar{\alpha}_{t_i}}}\right) + \sqrt{1-\bar{\alpha}_{t_{i-1}}}\cdot\epsilon_\theta\left(\mathbf{x}_{t_i}, t_i\right). \tag{20}$$

The algorithm 2 illustrates the fast and deterministic process of generating stego images. In essence, the sender begins by embedding the secret message, resulting in a stego latent $\mathbf{z}_s$, which is then used as the initial noise input for the StegoDiffusion model. Through accelerated denoising steps $p_\theta(\mathbf{x}_{t_{i-1}} \mid \mathbf{x}_{t_i})$, a clean image $\mathbf{x}_0$ is gradually obtained. Finally, after quantization, the image $\mathbf{x}_0$ containing floating-point data is converted to a digital image $\mathbf{x}^s$ with pixel values that vary between 0 and 255. This quantized image $\mathbf{x}^s$ can be stored and read on the hardware.

Algorithm 2 Generation of Stego Images

```
Data: Trained StegoDiffusion model; Fast sampling
      sequence of generation steps
      $\{t_1, \cdots, t_i, \cdots, t_T\} \subseteq \{1, 2, \cdots, \phi\}$;
      Hyper-parameters $\bar{\alpha}_t$ for each step ($t \in [1, \phi]$).
Result: The set of stego images.
1 while in generation steps do
2     Input the $n$ bits secret message sequence $\mathbf{d}$, where
      $\mathbf{d} \in \{0,1\}^n$;
3     Embed $\mathbf{d}$ to the latent $\mathbf{z}$ frequency domain to get the
      stego latent $\mathbf{z}_s$;
4     Set $\mathbf{x}_T$ equal to $\mathbf{z}_s$, $\mathbf{x}_T := \mathbf{z}_s$, as starting noise for the
      StegoDiffusion model generation process ;
5     Denoise $\mathbf{x}_T$ to obtain clean image $\mathbf{x}_0$, by fast
      generation process $p_\theta(\mathbf{x}_{t_{i-1}} \mid \mathbf{x}_{t_i})$;
6     Quantize $\mathbf{x}_0$ results in the final stego image $\mathbf{x}^s$, which
      is then saved in PNG format.
```

Algorithm 3 Secret Data Extraction

```
Data: The shared StegoDiffusion model by the sender,
      fast sampling sequence
      $\tau = \{t_1, \cdots, t_i, \cdots, t_T\} \subseteq \{1, 2, \cdots, \phi\}$,
      hyper-parameters $\bar{\alpha}_t$ for each step ($t \in [1, \phi]$), and
      the received stego images.
Result: The extracted secret data.
1 while in extraction steps do
2     Input a stego image $\mathbf{x}^s$ with integer pixel values;
3     Convert $\mathbf{x}^s$ into floating-point format;
4     Use the diffusion process $q(\mathbf{x}_{t_{i+1}} \mid \mathbf{x}_{t_i})$ to recover the
      original image sequence $\mathbf{x}^s \to \mathbf{x}_0 \cdots \to \mathbf{x}_T$;
5     Set reconstructed stego latent $\mathbf{z}'_s$ equal to $\mathbf{x}_T$,
      $\mathbf{z}'_s := \mathbf{x}_T$;
6     Retrive secret data $\mathbf{d}'$ from $\mathbf{z}'_s$ by Eqs. 22-23.
```

D. Secret Data Extraction

Similar to stego image generation, the process of data extraction also employs accelerated sampling methods. StegoDiffusion is reversible, and based on the accelerated generation process described in Eq. 20, the transition probability for the accelerated diffusion process in StegoDiffusion, $q(\mathbf{x}_{t_{i+1}} \mid \mathbf{x}_{t_i})$, can be derived as:

$$\mathbf{x}_{t_{i+1}} = \sqrt{\bar{\alpha}_{t_{i+1}}}\left(\frac{\mathbf{x}_{t_i} - \sqrt{1-\bar{\alpha}_{t_i}}\epsilon_\theta\left(\mathbf{x}_{t_i}, t_i\right)}{\sqrt{\bar{\alpha}_{t_i}}}\right) + \sqrt{1-\bar{\alpha}_{t_{i+1}}}\cdot\epsilon_\theta\left(\mathbf{x}_{t_i}, t_i\right). \tag{21}$$

The procedure for recovering hidden secrets by the receiver is illustrated in Algorithm 3. With the shared data extraction

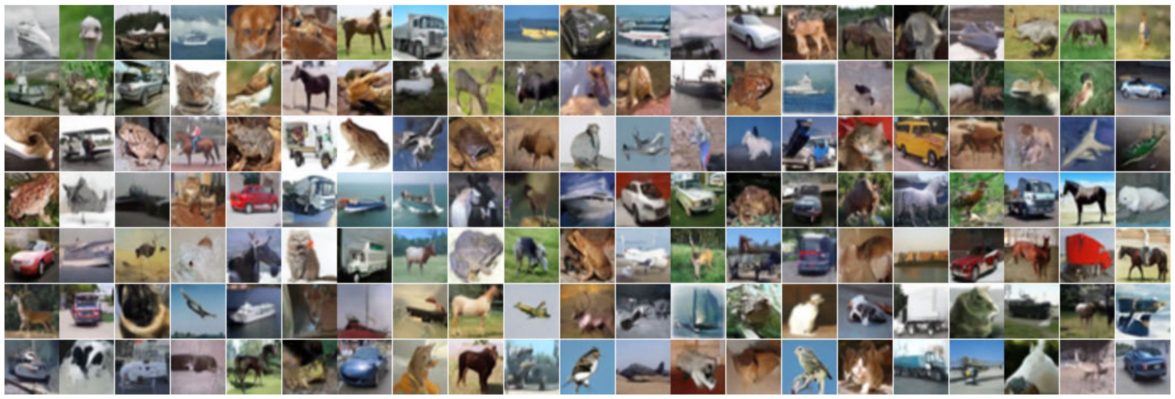


Fig. 4. Stego landscape images of size 32 × 32 generated by the GSD trained on Cifar10 dataset. With the parameter T=50, the payload is 1bpp, Acc is 99. 91%, the value of Pe is 0.4574, and the value of Fid is 7.57.

method and the StegoDiffusion model provided by the sender, the recipient can accurately recover the hidden data from the stego image $\mathbf{x}^s$. Specifically, the stego image $\mathbf{x}^s$ with integer pixel values is first converted to floating point data for neural network processing. Then, through the T-step noise injection process of StegoDiffusion, the clean image is transformed into a noisy image $\mathbf{x}^s \rightarrow \mathbf{x}_0 \rightarrow \mathbf{x}_1 \cdots \rightarrow \mathbf{x}_{T-1} \rightarrow \mathbf{z}'_s := \mathbf{x}_T$. At this point, the noisy image $\mathbf{x}_T$ serves as the retrived stego latent $\mathbf{z}'_s$. The final recovered secrets $\mathbf{d}'$ can then be extracted from $\mathbf{z}'_s$ according to the extraction method:

$$\mathbf{Coeff}_r = \mathrm{DCT}(\mathbf{z}'_s), \tag{22}$$

$$\mathbf{d}' = \mathrm{reshape}\left(\left\lceil \frac{\mathrm{Sign}(\mathbf{Coeff}_r) + 1}{2} \right\rceil\right), \tag{23}$$

where DCT($\cdot$) represents performing a discrete cosine transformation on the input matrix, $\mathbf{Coeff}_r$ represents the frequency domain coefficients obtained from DCT transformation of the recovered stego latent $\mathbf{z}'_s$, Sign($\cdot$) computes the sign matrix of the input, which contains $+1$ or -1 values, $\lceil\cdot\rceil$ performs the upward rounding operation on the input data, and reshape($\cdot$) adjusts the input matrix into a sequence form, ensuring that the output $\mathbf{d}'$ and hidden $\mathbf{d}$ have the same shape.

IV. Experimental Results

A. Details of Implementation

The experiments are conducted using Ubuntu 18.04, PyTorch 1.8 and an Nvidia 3090 GPU. Before training StegoDiffusion, the hyperparameters are set as follows: maximum diffusion step $\phi = 1000$, the coefficient for each step t is $\bar{\alpha}_t = \prod_{i=1}^{t}(1 - \frac{0.02i}{\phi})$, where $t \in [1, \phi]$. During StegoDiffusion training, the learning rate of the Adam optimizer is 4e-2. After training, the sender accelerates the sampling to generate stego images based on a sub-step $\tau = \{t_1, t_2, \cdots, t_{i-1}, t_i, \cdots, t_T\}$, where the sampling steps $T \leq \phi$. The sampling interval, denoted as Δt, is defined as $\Delta t = t_i - t_{i-1} = \frac{\phi}{T}$. Various sampling steps T result in different sampling intervals. Our experiments test the performance of the model under different sampling steps T.

In experiments, we conduct model training on the CelebA [62], Cifar10 [63], and Lsun-Bedroom [64] datasets, to generate 64×64 stego face images, 32×32 stego landscape images, and 64×64 stego bedroom images, respectively. The model undergoes training with an extensive dataset of around 130 million images for each dataset.

B. Metrics

Evaluation of the GSD scheme involves five key metrics: bpp, Acc, Fid, Pe Time, and Pe. These metrics represent different aspects of the model's performance: payload of secrets, visual quality of stego image, extraction accuracy, security, and time efficiency, respectively.

The bpp metric measures the quantity of secret data bits per pixel in a single channel, calculated as:

$$\mathrm{bpp} = \frac{\mathrm{len}(\mathbf{d})}{C \times H \times W}, \tag{24}$$

where $\mathbf{d}$ is a binary sequency containing secret data; len($\cdot$) calculates the bit count in the sequence $\mathbf{d}$; the variable C denote the channel numbers in the stego image (3 for a color image and 1 for a grayscale image), the variables H and W are the height and width of the stego image.

The Fid (Fréchet Inception Distance) metric is utilized to assess the quality of images, defined as:

$$\mathrm{Fid} = \left\| \nu_r - \nu_g \right\|^2 + \mathrm{Tr}\left(\omega_r + \omega_g - 2\left(\omega_r \omega_q\right)^{1/2}\right), \tag{25}$$

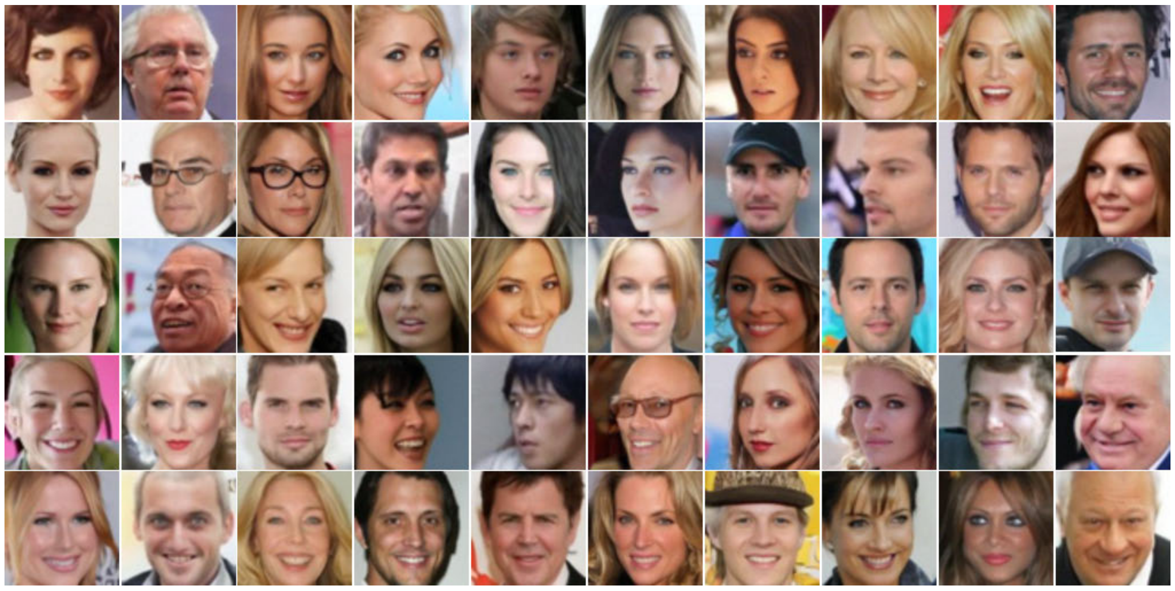
where ν and ω respectively denote the feature mean and covariance, where subscripts r and g indicate that the features are extracted from real and generated images, respectively.

The Acc metric indicates the extraction accuracy, defined as:

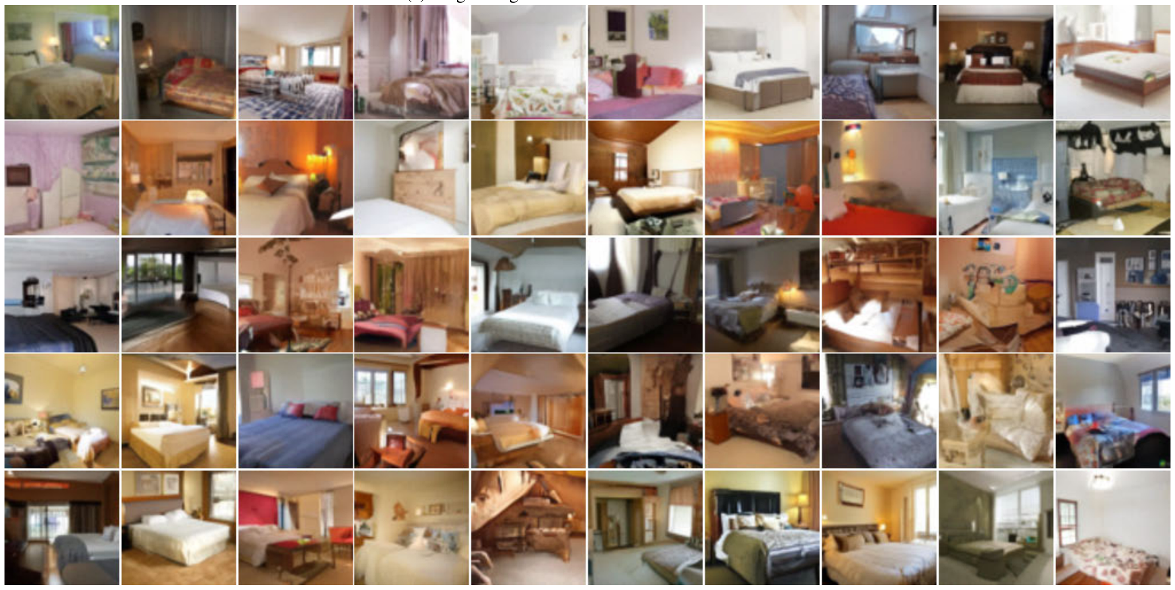
$$\mathrm{Acc} = \frac{\mathbf{d} \odot \mathbf{d}'}{\mathrm{len}(\mathbf{d})}, \tag{26}$$

where $\mathbf{d}/\mathbf{d}'$ is the embeded/recovered secret data sequence. There may be differences between $\mathbf{d}$ and $\mathbf{d}'$. The symbol $\odot$ signifies the exclusive NOR (XNOR) operation in the element.

Pe represents the mean error rate in classification when using a steganalysis detector for binary classification of positive and negative samples, and it is used to measure the security of the steganographic algorithms. In our scheme, the positive samples are the generated cover images without secret



(a) Stego images of face on the CelebA dataset



(b) Stego images of bedroom on the Bedroom dataset

Fig. 5. Stego images of size 64 × 64 by the GSD model trained on different datasets. For the CelebA dataset with the parameter T=50: the payload is 1 bpp, Acc is 100.00%, the value of Pe is 0.4850, and the value of Fid is 8.10; for the Bedroom dataset with the parameter T=50: the payload is 1 bpp, Acc is 99.93%, the value of Pe is 0.4620 and the value of Fid is 11.38.

data, and the negative samples are the generated stego images with secret data.

$$\mathrm{Pe} = \min_{P_{FA}} \frac{1}{2}(P_{FA} + P_{MD}), \tag{27}$$

where P_{FA} stands for the rate of false alarms, while P_{MD} indicates the rate of miss-detections in steganalysis detection. The Pe metric varies between 0 and 1, and an ideal value of 0.5 indicates that the probability of correctly distinguishing between a stego and a cover image is as low as random guessing, thus maximizing the security of the steganographic system.

The Time metric indicates the average duration for generating stego images or extracting hidden data, computed as:

$$\mathrm{Time} = \frac{t_{gen} + t_{ext}}{2}, \tag{28}$$

here, t denotes the time taken, with subscripts gen and ext representing the generation and extraction processes, respectively. It is crucial to emphasize that the time calculation is based on each individual image, so the unit of time is in seconds per image.

C. Effectiveness

We trained GSD on the Cifar10 dataset, and the trained GSD model can produce stego landscape images with a data payload of 1 bit per pixel (bpp) and a size of 32×32, as shown in Fig. 4, which illustrates 10 classes of randomly generated landscape

TABLE I
THE OVERALL EFFECTIVENESS OF THE GSD SCHEME WITH VARIOUS VALUES OF T AND DATASETS

Dataset	T	Acc(%) ↑	Pe→ 0.5	Fid↓	Time↓
Cifar10 32×32	10	99.66	0.4552	15.34	7.33e-3
	50	99.91	0.4574	7.57	3.62e-2
	100	99.93	0.4595	7.45	0.0861
	200	99.93	0.4664	8.08	0.1605
	500	99.94	0.4796	9.10	0.3509
CelebA 64×64	10	99.84	0.4743	15.77	4.02e-2
	50	100.00	0.4850	8.10	0.1346
	100	100.00	0.4575	9.28	0.2520
	200	100.00	0.4967	11.61	0.4893
	500	100.00	0.4781	13.91	1.2071
Bedroom 64×64	10	99.51	0.4530	17.24	1.78e-2
	50	99.93	0.4620	11.38	0.1094
	100	99.98	0.4698	12.46	0.2128
	200	99.99	0.4647	14.20	0.4684
	500	100.00	0.4730	16.17	0.9954

images. When GSD is trained on the CelebA or Lsun-bedroom datasets, it can generate stego face images or bedroom images with a payload of 1 bpp and a size of 64 × 64, as depicted in Figs. 5a and 5b, respectively. The generated stego images appear slightly blurry because the image resolution and size affect the image clarity. As the size of the images that the model can generate increases, the clarity of the generated stego images also improves. Hence, it is advisable to assess image quality using the Fid metric instead of solely depending on human perception.

Table I presents the overall performance of the GSD scheme across various parameters and datasets. where the tunable parameter T represents the number of sampling steps during image generation. Because of the particular embedding method implemented in this scheme, the data payload of the stego images remains at a constant rate of 1 bpp. The performance on different datasets is quite impressive, with secret message extraction accuracy (Acc) approaching or equal to 100%, steganalysis detection error rates (Pe) detected by RICH detector [65] close to the ideal value of 0.5, image quality metric (Fid) around 10, and time metric (Time) indicating that the average generation/extraction time for one image is within 1 second.

Analyzing how different values of T impact the performance of GSD, as illustrated in Table I, it is clear that as T increases from 10 to 500, Acc approaches 100% and Pe approaches the optimal value of 0.5. However, the time taken for the generation/extraction process increases, and although the image quality initially improves, it eventually deteriorates. When T is between 50 and 100, all four metrics perform well, demonstrating excellent accuracy, security, image quality, and execution times for the scheme.

To further determine the optimal value of the sampling steps T, the model trained on the CelebA dataset is used to produce stego images with T values of 10, 50, 100, 200, 500, and 1000, respectively, corresponding to the images from the first row to the last row in Figure 6. It can be observed that only when T = 10 (first row), the content of the stego images differs slightly from those generated with other values of T. When T varies between 50 and 1000 (from the second row to the last row), the content of the stego images remains almost identical. Furthermore, considering that larger values of T result in slower image generation speeds, we conclude that the optimal sampling step number is $T = 50$.

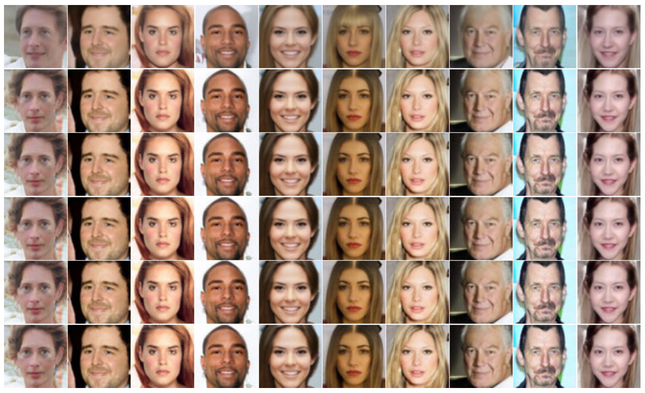


Fig. 6. Stego images generated by GSD using various sampling steps T. From the top row to the bottom row, the value of T in each row is 10, 50, 100, 200, 500, and 1000, respectively.

TABLE II
COMPARISON OF THE PROPOSED GSD SCHEME WITH SOTA METHODS

Method	bpp ↑	Acc (%) ↑	Fid ↓	Pe → 0.5
Liu [10]	4.17e-2	98.26	26.37	0.5350
Hu [11]	2.44e-2	91.73	30.81	0.4700
GSN [12]	0.33	98.15	12.88	0.4756
GSF [13]	1	99.96	22.45	0.4864
Proposed GSD	**1**	**100.00**	**8.10**	**0.4960**

D. Visualizing Reversibility

We illustrate the process of adding and removing noise during the generation of the stego image and the extraction of secret data in Fig. 7. For ease of illustration in the paper, we set the sampling step T = 10. During the stego image generation process, the stego latent $\mathbf{z}_s$ undergoes a 10-step denoising process, $\mathbf{x}_{10} := \mathbf{z}_s \to \mathbf{x}_9 \to \mathbf{x}_8 \cdots \to \mathbf{x}_1 \to \mathbf{x}_0$, resulting in natural stego images, as shown in the first, third and fifth rows from left to right in Fig. 7. In contrast, during secret data extraction, a noise addition process is applied to the quantized PNG images for ten steps, $\mathbf{x}_0 := \mathbf{x}^s \to \mathbf{x}_1 \to \mathbf{x}_2 \cdots \to \mathbf{x}_9 \to \mathbf{z}'_s := \mathbf{x}_{10}$, resulting in the retrive of the stego latent $\mathbf{z}'_s$. This process is illustrated in the second, fourth, and sixth rows from right to left in Fig. 7. The reconstrcted stego latent $\mathbf{z}'_s$ closely resembles the original input stego latent $\mathbf{z}_s$, with an absolute difference between them less than 1e-2. Combined with the secret data extraction method in Eqs. 22 and 23, the accuracy of secret data extraction can be close to or equal to 100%, demonstrating the reversibility of the GSD scheme.

E. Performance Comparison

In this part, we evaluate the superiority of GSD compared to other state-of-the-art (SOTA) steganography schemes,

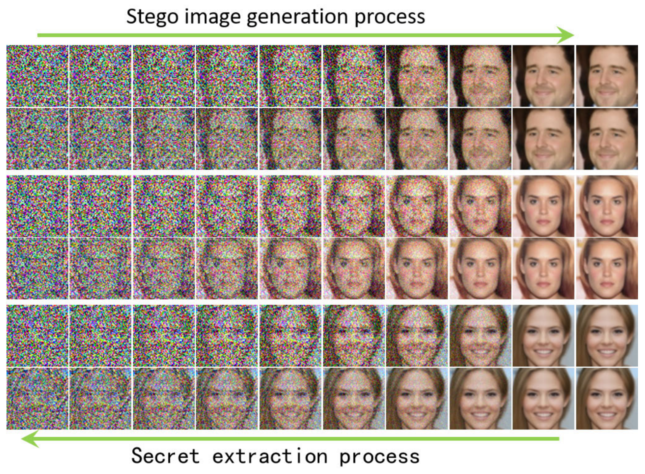


Fig. 7. Stepwise generation of stego images and extraction of secret data in GSD. Rows one, three, and five depict the progressive denoising process for stego image generation from the stego latent $\mathbf{z}_s$, while rows two, four, and six illustrate the incremental restoration process of $\mathbf{z}_s$ from the stego image.

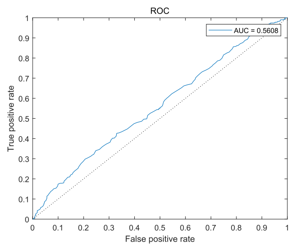
as shown in Table II. The methods proposed by Liu et al. [10], Hu et al. [11], and GSN [12] listed in Table II are all steganography methods that generate content using a generative adversarial network (GAN). GSF [13] utilizes the Flow model to generate stego images. GSD is the generative steganography method we propose, which is founded on the diffusion model. In comparative experiments, our method uses the optimal sampling step $T = 50$.

We conduct comparative experiments on the CelebA dataset, where we evaluate the secret data payload (bpp), the accuracy of secret data extraction (Acc), the quality of the image (Fid) and the security (Pe) of each method for comparison. The payload differs based on the data embedding technique employed by each method. When extracting secret data, stego images generated by each method are first saved as PNG images, from which secret data is extracted. Fid is calculated using Eq. 25. To assess the security of the methods, the steganalysis network YeNet [66] is first trained on 5000 stego/cover images generated by each method. Then, the steganalysis network that has been trained is applied to identify 1000 stego/cover images produced by each technique. The detection error rate, denoted as Pe and described in Eq. 27, is then calculated. It is important to mention that the cover images utilized for steganalysis detection are created directly from randomly selected Gaussian latent $\mathbf{z}$ without any embedded secret message.

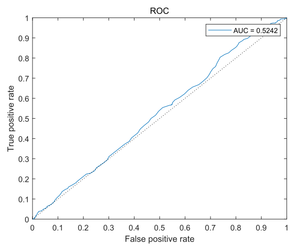
There is a trade-off among the four metrics of steganography methods, where increasing the payload can lead to decreases in extraction accuracy, image quality, and security performance. Methods such as Liu et al. [10] and Hu et al. [11] reduce the payload of secret data to achieve higher extraction accuracy and better image quality. For these two methods, the payload typically ranges around 1e-2 and the FID value is around 30, indicating high-quality images at that time. On the contrary, GSN [12] and GSF [13] improve the payload capacity while maintaining high extraction precision, image quality, and security. In steganography schemes, we consider extraction accuracy to be a critical metric. If hidden data

TABLE III
Pe Values of Generated Images by Different Steganalysis Methods

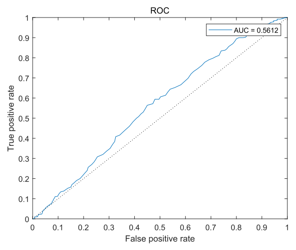
Dataset	RICH[65]	YeNet[66]	StegNet[67]	ZhuNet[68]	SiaStegNet[69]
CelebA	0.4575	0.4960	0.4965	0.4075	0.5005
Bedroom	0.4698	0.4972	0.4910	0.4465	0.4815



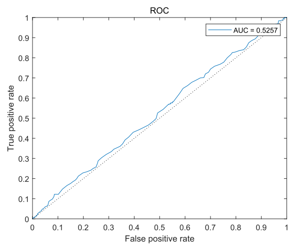
(a) $T = 50$



(b) $T = 100$



(c) $T = 200$



(d) $T = 1000$

Fig. 8. Detection ROC curves at different diffusion steps for dataset Bedroom.

cannot be accurately extracted, even if it is not detected by steganalysis, covert communication would be considered unsuccessful. However, none of the above four methods can achieve the 100% extraction accuracy of hidden data from PNG stego images. From Table II, we can see that the GSD achieves 100% extraction accuracy while ensuring excellent payload, image quality, and security.

F. Security Analysis

In this part, we evaluate the security of the GSD method against different steganalysis detection methods using CelebA and Bedroom datasets. The steganalysis algorithms used include RICH [65], YeNet [66], StegNet [67], ZhuNet [68], and SiaStegNet [69]. The outcome of experiments are presented in Table III. For training each of the steganalysis networks, 5000 randomly generated images with/without secrets are used as cover/stego images, and training started from scratch with default parameters. After training, an additional 1000 cover/stego images are randomly generated for testing purpose, with sampling step $T = 100$. The Pe value represents the detection error rate on the test set. A Pe value approaching 0.5 indicates that the steganalysis algorithm is unable to differentiate between stego and cover images, indicating a high level of security. The GSD method performs well against steganalysis algorithms RICH, YeNet, StegNet, and SiaStegNet, while exhibiting slightly inferior performance against ZhuNet. This is because the secret data in GSD scheme is concealed within both low-frequency and high-frequency domain coefficients, whereas steganalysis algorithm ZhuNet

TABLE IV
THE TEST OF ROBUSTNESS AGAINST ATTACKS

Attack	Methods	Acc (%)
JPEG compression $Q = 90$	[51]	61
	GSD	60
Gaussian noise $n \sim N(0, 0.02^2)$	[51]	69
	GSD	99
Image sterilization	[51]	99
	GSD	98

employs multiple high-pass filtering and multi-scale pooling operations that effectively preserve information in the high-frequency domain. As a result, there is an increased probability of detecting the hidden data in high-frequency coefficients. However, the secret data in the low-frequency domain remains undetectable by ZhuNet, leading to a Pe value that does not fall around 0.5 but still within an acceptable range of 0.4 to 0.45.

Furthermore, for the Bedroom dataset, we plot the ROC curves against the detection of RICH [65] with different sampling steps T, as shown in Fig. 8. For a specific sampling step T, the RICH detector is initially trained on 5000 generated images containing both stego and cover images. Subsequently, the trained RICH is utilized to classify 1000 generated stego/cover images, calculating the true and false positive rates to generate the ROC curves. In steganalysis, the gray dashed line in Fig. 8 represents the ideal ROC curve, achieving an AUC value of 0.5. The experiments demonstrate that the GSD scheme exhibits excellent performance against RICH detection, indicating high security.

G. Robustness

To evaluate the robustness of GSD, we conduct tests on against JPEG compression, Gaussian noise, and image sterilization [70], which is designed to remove secret data from stego images. For a fair comparison on the CelebA dataset with a payload of 1 bpp, we also evaluated the robustness of [51], which, like our method, did not undergo robust training. The results are summarized in Table IV. Notably, both [51] and GSD exhibit limited robustness against JPEG compression, with extraction accuracy hovering around 60%. On the other hand, both [51] and GSD demonstrate robustness to image sterilization attack, achieving extraction accuracies of 99% and 98%, respectively. GSD also demonstrates notable robustness to Gaussian noise. Specifically, on a Gaussian noise attack $n \sim N(0, 0.02^2)$, our method achieves an extraction accuracy of 99%. Existing methods without robust training all perform poorly when faced with attacks, and the development of robust steganography is a key focus of our future work.

V. CONCLUSION

This paper proposes an improved generative steganography scheme based on the diffusion model. We investigate how to generate stego images by diffusion model while achieving exact extraction accuracy. Our scheme involves hiding secret data within the frequency domain to prevent secret data distortion. Besides, a diffusion model (StegDiffusion) is retrained to handle stego images containing hidden data. Our scheme demonstrates superiority over existing methods across all evaluation metrics. Due to current limitations in generative technology, we can only ensure the security of steganography by generating stego/cover images that are challenging to distinguish. A key focus for the future is improving the realism of the generated stego images, reducing both visual and statistical differences compared to real images.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable comments.

REFERENCES

[1] J. Tao, S. Li, X. Zhang, and Z. Wang, "Towards robust image steganography," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 2, pp. 594–600, Feb. 2019.
[2] P. C. Mandal, I. Mukherjee, G. Paul, and B. N. Chatterji, "Digital image steganography: A literature survey," *Inf. Sci.*, vol. 609, pp. 1451–1488, Sep. 2022.
[3] R. Patel, K. Lad, and M. Patel, "Study and investigation of video steganography over uncompressed and compressed domain: A comprehensive review," *Multimedia Syst.*, vol. 27, no. 5, pp. 985–1024, Oct. 2021.
[4] J. Kunhoth, N. Subramanian, S. Al-Maadeed, and A. Bouridane, "Video steganography: Recent advances and challenges," *Multimedia Tools Appl.*, vol. 82, no. 27, pp. 41943–41985, Nov. 2023.
[5] J. Wu, B. Chen, W. Luo, and Y. Fang, "Audio steganography based on iterative adversarial attacks against convolutional neural networks," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2282–2294, 2020.
[6] M. A. Majeed, R. Sulaiman, Z. Shukur, and M. K. Hasan, "A review on text steganography techniques," *Mathematics*, vol. 9, no. 21, p. 2829, Nov. 2021.
[7] N. Subramanian, O. Elharrouss, S. Al-Maadeed, and A. Bouridane, "Image steganography: A review of the recent advances," *IEEE Access*, vol. 9, pp. 23409–23423, 2021.
[8] N. Farooq and A. Selwal, "Image steganalysis using deep learning: A systematic review and open research challenges," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 6, pp. 7761–7793, Jun. 2023.
[9] M.-M. Liu, M.-Q. Zhang, J. Liu, Y.-N. Zhang, and Y. Ke, "Coverless information hiding based on generative adversarial networks," 2017, *arXiv:1712.06951*.
[10] X. Liu, Z. Ma, J. Ma, J. Zhang, G. Schaefer, and H. Fang, "Image disentanglement autoencoder for steganography without embedding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 2303–2312.
[11] D. Hu, L. Wang, W. Jiang, S. Zheng, and B. Li, "A novel image steganography method via deep convolutional generative adversarial networks," *IEEE Access*, vol. 6, pp. 38303–38314, 2018.
[12] P. Wei, S. Li, X. Zhang, G. Luo, Z. Qian, and Q. Zhou, "Generative steganography network," in *Proc. 30th ACM Int. Conf. Multimedia*, Oct. 2022, pp. 1621–1629.
[13] P. Wei, G. Luo, Q. Song, X. Zhang, Z. Qian, and S. Li, "Generative steganographic flow," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2022, pp. 1–6.
[14] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 6840–6851.
[15] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2020, *arXiv:2010.02502*.
[16] K. Yang, K. Chen, W. Zhang, and N. Yu, "Provably secure generative steganography based on autoregressive model," in *Proc. Int. Workshop Digit. Watermarking*. Cham, Switzerland: Springer, 2018, pp. 55–68.
[17] J. Ding, K. Chen, Y. Wang, N. Zhao, W. Zhang, and N. Yu, "Discop: Provably secure steganography in practice based on," in *Proc. IEEE Symp. Secur. Privacy (SP)*, Mar. 2023, pp. 2238–2255.
[18] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Apr. 2022, pp. 10684–10695.

[19] N. Provos and P. Honeyman, "Hide and seek: An introduction to steganography," *IEEE Security Privacy*, vol. 1, no. 3, pp. 32–44, May/Jun. 2003.

[20] T. Sharp, "An implementation of key-based digital signal steganography," in *Proc. Int. Workshop Inf. Hiding*, vol. 2137. Cham, Switzerland: Springer, 2001, pp. 13–26.

[21] L. M. Marvel, C. G. Boncelet, and C. T. Retter, "Spread spectrum image steganography," *IEEE Trans. Image Process.*, vol. 8, no. 8, pp. 1075–1083, Aug. 1999.

[22] J. Fridrich and M. Goljan, "Digital image steganography using stochastic modulation," *Proc. SPIE*, vol. 5020, pp. 191–202, Jun. 2003.

[23] D. Upham, "Jsteg steganographic algorithm," 1999. [Online]. Available: https://www.nic.funet.fi/pub/crypt/steganography/

[24] A. Westfeld, "F5—A steganographic algorithm," in *Proc. Int. Workshop Inf. Hiding*. Cham, Switzerland: Springer, Jan. 2001, pp. 289–302.

[25] J. Fridrich, T. Pevnỳ, and J. Kodovskỳ, "Statistically undetectable JPEG steganography: Dead ends challenges, and opportunities," in *Proc. 9th Workshop Multimedia Secur.*, 2007, pp. 3–14.

[26] N. Provos, "Defending against statistical steganalysis," in *Proc. 10th USENIX Secur. Symp. (USENIX Secur. 01)*, vol. 10, Aug. 2001, p. 24.

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

[28] T. Pevný, T. Filler, and P. Bas, "Using high-dimensional image models to perform highly undetectable steganography," in *Proc. Int. Workshop Inf. Hiding*. Cham, Switzerland: Springer, 2010, pp. 161–177.

[29] V. Holub and J. Fridrich, "Digital image steganography using universal distortion," in *Proc. 1st ACM Workshop Inf. Hiding Multimedia Secur.*, Montpellier, France, Jun. 2013, pp. 59–68.

[30] B. Li, M. Wang, J. Huang, and X. Li, "A new cost function for spatial image steganography," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 4206–4210.

[31] V. Sedighi, R. Cogranne, and J. Fridrich, "Content-adaptive steganography by minimizing statistical detectability," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 221–234, Feb. 2016.

[32] M. A. Wani and B. Sultan, "Deep learning based image steganography: A review," *WIREs Data Mining Knowl. Discovery*, vol. 13, no. 3, p. 1481, May 2023.

[33] W. Tang, S. Tan, B. Li, and J. Huang, "Automatic steganographic distortion learning using a generative adversarial network," *IEEE Signal Process. Lett.*, vol. 24, no. 10, pp. 1547–1551, Oct. 2017.

[34] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, "Hidden: Hiding data with deep networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 657–672.

[35] J. Hayes and G. Danezis, "Generating steganographic images via adversarial training," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 1951–1960.

[36] M. Tancik, B. Mildenhall, and R. Ng, "StegaStamp: Invisible hyperlinks in physical photographs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2117–2126.

[37] R. Zhang, S. Dong, and J. Liu, "Invisible steganography via generative adversarial networks," *Multimedia Tools Appl.*, vol. 78, no. 7, pp. 8559–8575, Apr. 2019.

[38] W. Tang, B. Li, M. Barni, J. Li, and J. Huang, "An automatic cost learning framework for image steganography using deep reinforcement learning," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 952–967, 2021.

[39] C. Yu, D. Hu, S. Zheng, W. Jiang, M. Li, and Z.-Q. Zhao, "An improved steganography without embedding based on attention GAN," *Peer Peer Netw. Appl.*, vol. 14, no. 3, pp. 1446–1457, May 2021.

[40] J. Yang, D. Ruan, J. Huang, X. Kang, and Y.-Q. Shi, "An embedding cost learning framework using gan," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 839–851, 2019.

[41] Y. Yao, L. Huang, H. Wang, Q. Chang, Y. Ren, and F. Xiao, "Robust adaptive steganography based on adaptive STC-ECC," *IEEE Trans. Multimedia*, vol. 26, pp. 5477–5489, 2024.

[42] J. Jing, X. Deng, M. Xu, J. Wang, and Z. Guan, "HiNet: Deep image hiding by invertible network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 4733–4742.

[43] S.-P. Lu, R. Wang, T. Zhong, and P. L. Rosin, "Large-capacity image steganography based on invertible neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10816–10825.

[44] S. Baluja, "Hiding images within images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, pp. 1685–1697, Jul. 2020.

[45] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2642–2651.

[46] Z. Zhang, G. Fu, R. Ni, J. Liu, and X. Yang, "A generative method for steganography by cover synthesis with auxiliary semantics," *Tsinghua Sci. Technol.*, vol. 25, no. 4, pp. 516–527, Aug. 2020.

[47] W. Jiang, D. Hu, C. Yu, M. Li, and Z.-Q. Zhao, "A new steganography without embedding based on adversarial training," in *Proc. ACM Turing Celebration Conf.-China*, May 2020, pp. 219–223.

[48] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.

[49] J. Li et al., "A generative steganography method based on wgan-gp," in *Proc. 6th Int. Conf. Artif. Intell. Secur. (ICAIS)*, Hohhot, China. Cham, Switzerland: Springer, 2020, pp. 386–397.

[50] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 5769–5779.

[51] Z. Zhou et al., "Secret-to-Image reversible transformation for generative steganography," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 5, pp. 4118–4134, Sep. 2023.

[52] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1×1 convolutions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, Jan. 2018, pp. 10215–10224.

[53] Y. Peng, D. Hu, Y. Wang, K. Chen, G. Pei, and W. Zhang, "StegaDDPM: Generative image steganography based on denoising diffusion probabilistic model," in *Proc. 31st ACM Int. Conf. Multimedia*, Oct. 2023, pp. 7143–7151.

[54] L. Yang et al., "Diffusion models: A comprehensive survey of methods and applications," 2022, *arXiv:2209.00796*.

[55] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 9, pp. 10850–10869b, Mar. 2023.

[56] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 34, 2021, pp. 8780–8794.

[57] A. Nichol et al., "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1–12.

[58] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with CLIP latents," 2022, *arXiv:2204.06125*.

[59] C. Saharia et al., "Photorealistic text-to-image diffusion models with deep language understanding," 2022, *arXiv:2205.11487*.

[60] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, vol. 139, 2021, pp. 8748–8763.

[61] N. Metropolis and S. Ulam, "The Monte Carlo method," *J. Amer. Statist. Assoc.*, vol. 44, no. 247, pp. 335–341, 1949.

[62] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3730–3738.

[63] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[64] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop," 2015, *arXiv:1506.03365*.

[65] M. Goljan, J. Fridrich, and R. Cogranne, "Rich model for steganalysis of color images," in *Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*, Dec. 2014, pp. 185–190.

[66] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2545–2557, Nov. 2017.

[67] X. Deng, B. Chen, W. Luo, and D. Luo, "Fast and effective global covariance pooling network for image steganalysis," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, Jul. 2019, pp. 230–234.

[68] R. Zhang, F. Zhu, J. Liu, and G. Liu, "Depth-wise separable convolutions and multi-level pooling for an efficient spatial CNN-based steganalysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1138–1150, 2020.

[69] W. You, H. Zhang, and X. Zhao, "A Siamese CNN for image steganalysis," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 291–306, 2021.

[70] M. Imon, P. Goutam, and A. J. Jawahar, "Defeating steganography with multibit sterilization using pixel eccentricity," *IPSI BgD Trans. Adv. Res.*, vol. 11, no. 1, pp. 25–34, 2015.

Qing Zhou received the B.S. degree in electronic information engineering from Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu, China, in 2016, and the M.S. degree in signal and information processing from the University of Shanghai for Science and Technology, China, in 2019. She is currently pursuing the Ph.D. degree with the School of Computer Science, Fudan University, China. Her research interests include image processing, data hiding, and steganography.

Ping Wei received the B.S. degree in mechanical engineering from Central South University, Changsha, China, in 2011, the M.S. degree in mechanical engineering from Sichuan University, Chengdu, China, in 2017, and the Ph.D. degree in computer science from Fudan University, Shanghai, China. He is currently a Lecturer with Yunnan University, Kunming, China. His research interests include multimedia security, data hiding, computer vision, NLP, and machine learning.

Zhenxing Qian (Senior Member, IEEE) received the B.S. and Ph.D. degrees from the University of Science and Technology of China (USTC) in 2003 and 2008, respectively. He is currently a Professor with the School of Computer Science, Fudan University. He is also the Vice Dean of the Key Laboratory of China Culture and Tourism Ministry. Until now, he has published over 220 peer-reviewed papers, many of which are published on IEEE TRANSACTIONS and conferences, such as CVPR, ICCV, NerurIPS, AAAI, IJCAI, and ACM MM. His research interests include multimedia security, AI security, and intelligence of culture and tourism. He serves as an Associate Editor for IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE TRANSACTIONS ON AUDIO, SPEECH AND LANGUAGE PROCESSING, *Signal Processing*, and *Journal of Visual Communication and Image Representation*.

Xinpeng Zhang (Senior Member, IEEE) received the B.S. degree in computational mathematics from Jilin University, China, in 1995, and the M.E. and Ph.D. degrees in communication and information system from Shanghai University, China, in 2001 and 2004, respectively. He is currently a Professor with the School of Computer Science, Fudan University. His research interests include information hiding, image processing, and digital forensics. He has published over 200 articles in these areas.

Sheng Li (Member, IEEE) received the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2013. From 2013 to 2016, he was a Research Fellow with the Rapid Rich Object Search (ROSE) Laboratory, Nanyang Technological University. He is currently an Associate Professor with the School of Computer Science, Fudan University, China. His research interests include biometric template protection, pattern recognition, and multimedia forensics and security. He was a recipient of the IEEE WIFS Best Student Paper Silver Award.