GVIS: Generative Vector Image Steganography

Zihao Xu
Changchun University
230702293@mails.ccu.edu.cn

Dawei Xu
Changchun University
xudw@ccu.edu.cn

Zihan Li
Beijing Institute of Technology
zihanl@bit.edu.cn

Xixi Zheng
The Hong Kong Polytechnic University
xi-xi.zheng@polyu.edu.hk

Chuan Zhang†
Beijing Institute of Technology
chuanz@bit.edu.cn

Abstract

*Vector images have attracted increasing attention in the field of information hiding in recent years due to their scalability and structural properties. However, existing steganographic methods for vector images often introduce noticeable modifications to the files themselves, resulting in potential security risks and limited embedding capacity. Motivated by recent advances in diffusion models and image generative steganography, we propose GVIS, a novel **G**enerative **V**ector **I**mage **S**teganography framework. GVIS deterministically generates raster images using diffusion models, which are subsequently vectorized into vector images. On the sender side, we design a lightweight overlap detection algorithm to identify cubic Bézier curve control points suitable for data embedding, which enables the secret information to be encoded into the coordinate parameters of these control points. Then, the receiver can use the pre-shared settings to reconstruct the generation process and accurate message extraction by difference. Extensive theoretical analysis and experimental results demonstrate that GVIS achieves high-capacity, high-accuracy, secure, and training-free steganography. To the best of our knowledge, this is the first attempt to apply generative models to vector image steganography.*

1. Introduction

Steganography, the practice of hiding information within digital media, plays an important role in secure communication by concealing the existence of the message. With growing surveillance and interception threats, the need for covert, high-capacity, and robust steganographic techniques has become increasingly important. Traditional steganographic methods based on hand-crafted features or simple transforms are increasingly vulnerable to detection and performance degradation under common processing operations [5]. Therefore, developing high-fidelity and robust steganographic methods is critical for modern data privacy and security.

Among various steganographic media, pixel-based images have been the most extensively explored. Classic Least Significant Bit (LSB) techniques [28], although simple and efficient, are highly susceptible to statistical analysis and common image processing operations. To address these limitations, frequency domain approaches distribute hidden information across transformed coefficients, offering improved resistance to compression and filtering [1, 21]. Recently, deep learning-based image steganography methods have emerged, jointly learning to embed and extract messages, thereby significantly enhancing secrecy and capacity [18, 34]. However, these approaches still rely heavily on existing cover images, suffer from limited flexibility, and remain vulnerable to steganalysis. Generative steganography addresses these limitations by embedding secret messages during data synthesis. By leveraging the expressive power of generative models, these methods synthesize realistic images that can directly serve as steganographic carriers. For example, flow-based models exploit invertibility for lossless message embedding [31], GAN-based methods generate high-fidelity carriers without modifying existing media [16], and diffusion-based approaches embed messages in latent space for improved robustness and capacity [39]. However, the majority of current approaches are restricted to pixel-domain images, constraining both capacity and robustness against distortion.

In contrast, vector images offer a compelling alternative as steganographic carriers, thanks to their resolution independence, compact representation, and inherent lossless nature. However, existing vector-based steganographic methods are typically limited by low embedding capacity and poor practicality [20, 24]. In this work, we introduce GVIS,

†Corresponding author

to the best of our knowledge, the first generative vector image steganography framework. GVIS utilizes a deterministic diffusion model and a lightweight vectorization algorithm to progressively generate steganographic vector images. By integrating the flexibility and high capacity of generative steganography with the compactness and fidelity of vector images, GVIS offers a training-free and robust solution. Specifically, the sender uses a textual prompt and random seed to generate a pixel image, which is then converted into a vector image via a lightweight vectorization process. A dedicated encoding strategy embeds the secret message into the vector image without degrading its visual quality or causing noticeable statistical distortion. The stego-vector image is made public, while the prompt and seed are shared through a secure channel. The receiver regenerates the same vector image from the shared information and decodes the hidden message by computing the residual between the regenerated image and the received stego image.

Our main contributions are summarized as follows:

- We propose GVIS, the first generative steganography framework for vector images, combining a deterministic diffusion model with lightweight vectorization for training-free and high-capacity stego generation.
- We design a novel message encoding scheme based on Bézier curve properties, preserving visual quality and improving message secrecy and statistical undetectability.
- Extensive experiments show that GVIS achieves superior vector quality, embedding capacity, and robustness, outperforming pixel-based generative methods.

2. Related Work

2.1. Pixel-Based Steganography

Pixel-based steganography has long been the dominant paradigm in digital information hiding. Traditional methods like Least Significant Bit (LSB) embedding [28] are simple and offer high embedding capacity, but are easily detected and fragile against image processing [5]. Frequency-domain techniques [1, 21] improve robustness by spreading information across transformed coefficients. Adaptive approaches further reduce distortion using handcrafted or learned embedding strategies [18, 34]. More recently, deep learning has enabled end-to-end message embedding and extraction, as demonstrated by methods like HiDDeN [44]. Generative steganography represents a new paradigm by embedding messages during image generation. Flow-based methods exploit model invertibility to enable lossless data embedding and recovery [31, 42]. GAN-based methods generate high-fidelity images that carry hidden data without modifying visible covers [16, 35, 43]. Diffusion-based methods embed information during the denoising process, offering strong robustness, imperceptibility, and controllability [9, 23, 39]. Despite these advances, pixel-based methods often suffer from relatively large file sizes and limited embedding capacity due to the inherent constraints of raster representations, which restricts their applicability in compact or high-volume steganographic scenarios.

2.2. Vector Image Steganography

Vector image steganography leverages the resolution-independent and compact representation of vector images for hidden data transmission. Compared with raster images, vector images offer lossless scalability and structural regularity, making them a promising yet underexplored steganographic medium. Early methods embed secrets by modifying the least significant digits of SVG path coordinates [2] or inserting control points into curves to encode bits while preserving visual appearance [20]. Other techniques segment Bézier curves and apply geometric perturbations to balance capacity and imperceptibility [15]. Recent advances include machine learning-based methods that modify low-level primitives (points, lines, curves) through learned encodings, enabling flexible, high-fidelity embedding while preserving renderability [24]. Despite progress, most existing techniques remain constrained by limited embedding capacity.

2.3. Vector Image Generation

Vector image generation has progressed through three main directions: differentiable rendering, text-to-SVG synthesis, and image-to-vector translation. Differentiable rendering [17] enables pixel-level supervision over vector primitives via gradient-based optimization. Text-to-SVG methods [7, 10, 32] leverage CLIP or diffusion models to synthesize SVG content from text prompts by optimizing or distilling primitives guided by text. Image-to-vector translation aims to convert raster images into SVGs, sometimes incorporating text for multimodal control. [19] introduces a layer-wise Bézier optimization framework for semantic vectorization, while [27] combines textual and visual inputs for stylized SVGs. [25] proposes a multimodal LLM that generates SVG code directly from image-text pairs. While these models offer high-quality and flexible outputs, they typically rely on large pretrained models and require substantial computation. In contrast, VTracer [29] is a utility to convert raster images into vector graphics, which provides a lightweight and deterministic solution.

3. Threat Model

We consider a privacy-oriented SVG-based hidden communication scenario. The sender publishes an SVG file through public channels, such as websites, social media platforms, or open file repositories. Its visible content, for example ordinary illustrations, icons, or decorative vector graphics, is designed to appear indistinguishable from benign public SVG files so as not to attract suspicion. Meanwhile, a

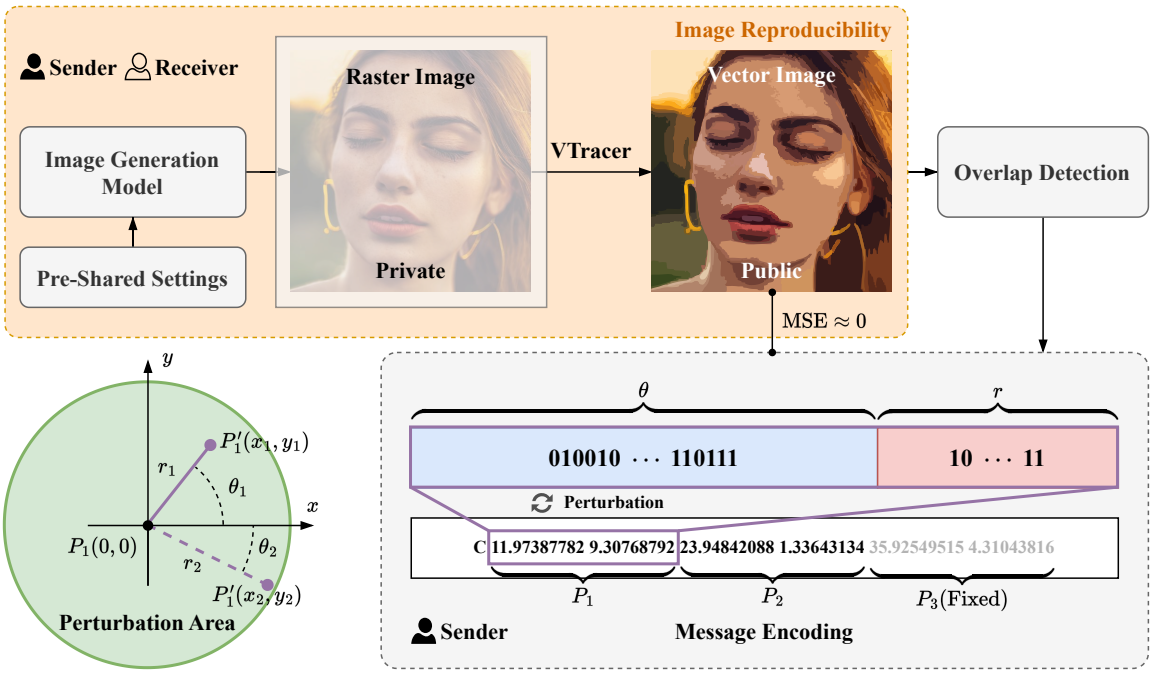


Figure 1. The overall framework of GVIS. The sender and receiver synchronize conditional information to generate images via a diffusion model, followed by vectorization using VTracer. The sender then employs a overlap detection algorithm to filter suitable curves and control points for steganography, and embeds information by perturbing these control points.

confidential message is embedded in the SVG and can only be recovered by authorized recipients. We primarily consider passive adversaries who can observe and analyze the published SVG but do not possess the secret seed required for reconstruction. Compared with raster images, SVG files are less likely to undergo lossy compression during transmission, although they may still experience transformations such as scaling, rendering variation, or format-preserving editing. The adversary aims to recover the hidden message by brute-forcing possible pseudo-random seeds, reverse-engineering the vector structure to identify hidden patterns, or analyzing metadata and geometric parameter anomalies, such as subtle path offsets or coordinate perturbations. We assume a black-box setting in which the adversary has no prior knowledge of the predefined pseudo-random number generator, seed length, or exact embedding strategy. Authorized recipients share the secret seed with the sender through a secure channel. Using this seed, they initialize the agreed pseudo-random number generator, regenerate the original source data of the SVG, and reconstruct the corresponding vector structure. The hidden message is then extracted by comparing the regenerated vector data with the received stego SVG.

4. Method

In this paper, we propose a framework for generative vector image steganography called GVIS, as illustrated in Figure 1. We first demonstrate the feasibility of GVIS through theoretical analysis. Next, we propose a lightweight overlap detection algorithm to mitigate potential steganalysis risks. Finally, we describe the complete embedding and extraction process.

4.1. Perturbation of Cubic Bézier Curves

Any SVG format vector image can be described as containing a large number of cubic Bézier curves. In many applications, such as animation, font design, and computer-aided design (CAD), precise manipulation of these control points is crucial. However, small perturbations to the control points do not significantly alter the curve shape, which makes them suitable for imperceptible information embedding.

4.1.1. Mathematical Definition of the Cubic Bézier Curve

A cubic Bézier curve is a parametric curve defined over the interval $t \in [0, 1]$, expressed as a weighted combination of four control points. Let $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3 \in \mathbb{R}^2$ denote the control points, where $\mathbf{P}_0$ and $\mathbf{P}_3$ are the fixed endpoints, and $\mathbf{P}_1$ and $\mathbf{P}_2$ are the adjustable control points. The curve $\mathbf{B}(t)$ is given by the following polynomial equation:

$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3. \quad (1)$$

Here, $\mathbf{B}(0) = \mathbf{P}_0$ and $\mathbf{B}(1) = \mathbf{P}_3$ are endpoints, while $\mathbf{P}_1$ and $\mathbf{P}_2$ determine the curve shape.

4.1.2. Perturbation for Control Points

To study the impact of control point adjustments, we introduce a perturbation model where the endpoints $\mathbf{P}_0$ and $\mathbf{P}_3$ remain stationary, and the control points $\mathbf{P}_1$ and $\mathbf{P}_2$ are subject to random displacements. Specifically, the perturbed control points are defined as:

$$\mathbf{P}_1' = \mathbf{P}_1 + \boldsymbol{\Delta}_1, \quad \mathbf{P}_2' = \mathbf{P}_2 + \boldsymbol{\Delta}_2, \tag{2}$$

where $\boldsymbol{\Delta}_1$ and $\boldsymbol{\Delta}_2$ are independent random vectors, each uniformly distributed within a disk of radius r centered at the origin (i.e., $\|\boldsymbol{\Delta}_1\| \leq r$ and $\|\boldsymbol{\Delta}_2\| \leq r$). The perturbed cubic Bézier curve $\mathbf{B}'(t)$ then becomes:

$$\mathbf{B}'(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1' + 3(1-t)t^2\mathbf{P}_2' + t^3\mathbf{P}_3. \tag{3}$$

4.1.3. Quantifying Curve Deviation via Mean Squared Error (MSE)

To quantify the deviation between $\mathbf{B}(t)$ and $\mathbf{B}'(t)$, we use the mean squared error (MSE):

$$\text{MSE} = \int_0^1 \|\mathbf{B}(t) - \mathbf{B}'(t)\|^2 \, dt. \tag{4}$$

Since $\boldsymbol{\Delta}_1$ and $\boldsymbol{\Delta}_2$ are random, we consider the expected value:

$$\mathbb{E}[\text{MSE}] = \int_0^1 \mathbb{E}[\|\mathbf{B}(t) - \mathbf{B}'(t)\|^2] \, dt. \tag{5}$$

This expectation accounts for all possible perturbations within the specified disks, offering a robust estimate of the curve's sensitivity to control point variations.

4.1.4. Analytical Derivation of the Expected MSE

We begin by calculating the difference $\delta\mathbf{B}(t) = \mathbf{B}'(t) - \mathbf{B}(t)$. Noting that $\mathbf{P}_0$ and $\mathbf{P}_3$ are unchanged, we obtain:

$$\begin{aligned}\delta\mathbf{B}(t) &= 3(1-t)^2 t(\mathbf{P}_1' - \mathbf{P}_1) + 3(1-t)t^2(\mathbf{P}_2' - \mathbf{P}_2) \\ &= 3(1-t)^2 t\boldsymbol{\Delta}_1 + 3(1-t)t^2\boldsymbol{\Delta}_2.\end{aligned} \tag{6}$$

Next, we compute the squared norm of the difference:

$$\begin{aligned}\|\delta\mathbf{B}(t)\|^2 &= \left\|3(1-t)^2 t\boldsymbol{\Delta}_1 + 3(1-t)t^2\boldsymbol{\Delta}_2\right\|^2 \\ &= 9(1-t)^4 t^2\|\boldsymbol{\Delta}_1\|^2 + 9(1-t)^2 t^4\|\boldsymbol{\Delta}_2\|^2 \\ &\quad + 18(1-t)^3 t^3(\boldsymbol{\Delta}_1 \cdot \boldsymbol{\Delta}_2).\end{aligned} \tag{7}$$

Here, the cross term arises from the dot product of the perturbation vectors.

Taking the expectation over the random perturbations, we leverage the independence and uniform distribution of $\boldsymbol{\Delta}_1$ and $\boldsymbol{\Delta}_2$. Due to symmetry in the disk, the expected dot product vanishes:

$$\mathbb{E}[\boldsymbol{\Delta}_1 \cdot \boldsymbol{\Delta}_2] = 0. \tag{8}$$

For a uniform distribution over a disk of radius r in two dimensions, the second moment of each coordinate can be explicitly calculated. The mean squared magnitude is given by

$$\mathbb{E}\left[\|\boldsymbol{\Delta}_i\|^2\right] = \frac{1}{\pi r^2}\int_0^r 2\pi\rho \cdot \rho^2 \, d\rho = \frac{r^2}{2}, \tag{9}$$

where the normalization constant is the area of the disk. Thus, the expected squared deviation at a given parameter value t can be written as:

$$\begin{aligned}\mathbb{E}\left[\|\delta\mathbf{B}(t)\|^2\right] &= 9(1-t)^4 t^2\mathbb{E}\|\boldsymbol{\Delta}_1\|^2 + 9(1-t)^2 t^4\mathbb{E}\|\boldsymbol{\Delta}_2\|^2 \\ &= \frac{9r^2}{2}\left[(1-t)^4 t^2 + (1-t)^2 t^4\right],\end{aligned} \tag{10}$$

making use of the fact that both control points are perturbed with identical, independent distributions. To obtain the overall mean squared error (MSE) for the entire curve, we integrate the expected squared deviation over the full parameter domain, yielding

$$\begin{aligned}\mathbb{E}[\text{MSE}] &= \int_0^1 \mathbb{E}\left[\|\delta\mathbf{B}(t)\|^2\right] dt \\ &= \frac{9r^2}{2}\int_0^1 (1-t)^4 t^2 dt + \frac{9r^2}{2}\int_0^1 (1-t)^2 t^4 dt \\ &= \frac{3}{35}r^2.\end{aligned} \tag{11}$$

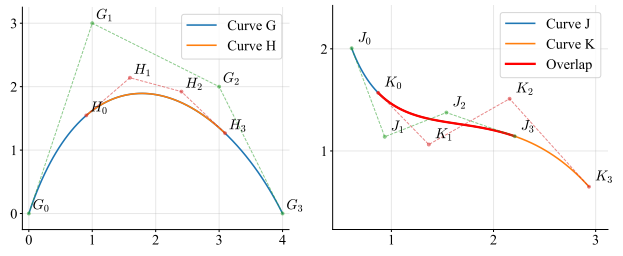
This result shows that the expected deviation is proportional to r^2 and independent of the absolute control point positions or the specific curve shape.

4.2. Overlap Detection

According to Theorem 5.9.1 in [22], if two C^∞ curve segments $r_1(t)$ and $r_2(\sigma)$ overlap over a finite interval, then they must coincide everywhere on that interval. If they do not coincide completely, the overlap can only terminate at a boundary point. Based on this observation, we further consider the following cases. In general, except for straight lines, when two curves share the same start and end points but differ in their control points, they do not mathematically overlap, even if they appear visually very close. Apart from the trivial case where the two curves are identical, Figure 2 illustrates two possible types of segment overlap. In essence, both cases correspond to curve segments obtained by subdividing the original curve. These subdivided Bézier segments are mutually independent: modifying one subdivided segment does not affect the others. The distinction between the two cases in Figure 2 is as follows: in Figure 2a, the overlapping portion, denoted by curve H, corresponds to only one subdivided segment of the original curve G, whereas in Figure 2b, curves J and K correspond to two different subdivided segments of the original curve.

Algorithm 1 Overlap detection

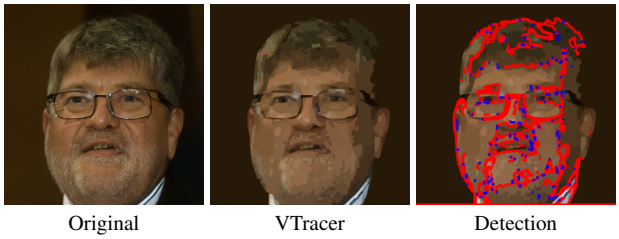
Input: Curve set $\mathcal{C} = \{C_1, \ldots, C_m\}$
Compute bounding boxes $\mathcal{B} = \{B_1, \ldots, B_m\}$
$\mathcal{O}_1 \leftarrow \emptyset$, $\mathcal{O}_2 \leftarrow \emptyset$
for each ordered pair (i, j), $i \neq j$ **do**
  if $B_i \cap B_j = \emptyset$ **then**
    continue
  end if
  $D \leftarrow C_i,\ A \leftarrow C_j$
  Case 1: complete overlap
  if D and A have matching endpoints and consistent control points **then**
    $\mathcal{O}_1 \leftarrow \mathcal{O}_1 \cup \{(i, j)\}$
  else if sampled points on D all lie on A **then**
    $\mathcal{O}_1 \leftarrow \mathcal{O}_1 \cup \{(i, j)\}$
  end if
  Case 2: partial overlap
  for each endpoint combination of D and A **do**
    if the two chosen endpoints lie on the opposite curves **then**
      if the sampled sub-curve of D lies on A **then**
        $\mathcal{O}_2 \leftarrow \mathcal{O}_2 \cup \{(i, j)\}$
        break
      end if
    end if
  end for
end for
Remove duplicate pairs from $\mathcal{O}_1$ and $\mathcal{O}_2$
return $\mathcal{O}_1, \mathcal{O}_2$



(a) Case 1 (b) Case 2

Figure 2. Two cases in which two cubic Bézier curves may overlap.

Based on the above theory, we believe that when two curves overlap, if the two cubic Bézier curves coincide almost everywhere, but the disturbance of the control points makes them slightly different, they may be vulnerable to steganalysis. Therefore, we need to exclude the following two cases of curve overlap: **Case 1.** The two endpoints (H_0, H_3) of a cubic Bézier curve H are on another cubic Bézier curve G and completely overlap with the curve G; **Case 2.** One endpoint (J_3) of a cubic Bézier curve J is on another cubic Bézier curve K and partly overlaps with the curve K. For case 1, first check whether the two endpoints of curve G coincide with those of curve H. If so, directly determine whether the control points coincide. Otherwise, uniformly sample points on curve H and determine whether all of them lie on curve G. For case 2, if endpoint J_3 of curve J lies on curve K and endpoint K_0 of curve K lies on curve J, then uniformly sample points on the segment of curve K between K_0 and J_3 and determine whether these points are on curve J. The practical effect of the overlap detection is shown in Figure 3. To further accelerate this process, we exploit the convex hull property of Bézier curves to accelerate the above process by using the bounding box (BBox) defined by the minimum and maximum x- and y-coordinates of the four control points of the curve as shown in Figure 4. The specific process is shown in Algorithm 1.



Original VTracer Detection

Figure 3. Visual examples of overlap detection results. In the third figure, red and blue are used to represent the first and second cases respectively.

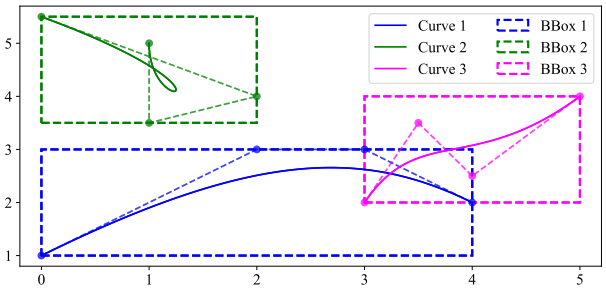


Figure 4. Visual example of bounding box acceleration detection process.

4.3. Message Hiding and Extraction

In this section, we will describe how to steganographically represent cubic Bézier curves. First, our method requires that cubic Bézier curves be reproducible. We found that, compared to most existing vector generation methods, such as [19, 32], which utilize DiffVG [17] and thus introduce randomness, diffusion-based image generation with fixed seeds and raster-to-vector conversion based on deterministic algorithms are well suited to steganography. Further-

Table 1. Quantitative comparison of our method with existing pixel-based and vector image steganography methods.

Dataset	Method	Type	SSIM	PSNR	Accuracy	Capacity (bits)
CelebA-HQ	DwtDctSvd [21]	Pixel	0.9861	40.22	0.9991	32
	RivaGAN [40]		0.9804	40.54	1.0000	32
	RoSteALS [4]		0.9087	31.01	0.9935	100
	StegoSVG [3]	Vector	0.9995	57.75	1.0000	48000
	StegoBIT [13]		0.9999	63.87	1.0000	6590
	svgsteg [11]		0.9999	61.63	1.0000	46441
	GVIS (Ours)		0.9999	67.87	1.0000	88878
LSUN-Bedrooms	DwtDctSvd [21]	Pixel	0.9850	40.05	1.0000	32
	RivaGAN [40]		0.9776	40.50	0.9994	32
	RoSteALS [4]		0.9188	30.91	0.9937	100
	StegoSVG [3]	Vector	0.9997	58.09	1.0000	41923
	StegoBIT [13]		0.9999	63.10	1.0000	6752
	svgsteg [11]		0.9999	62.96	1.0000	38484
	GVIS (Ours)		1.0000	69.50	1.0000	66830

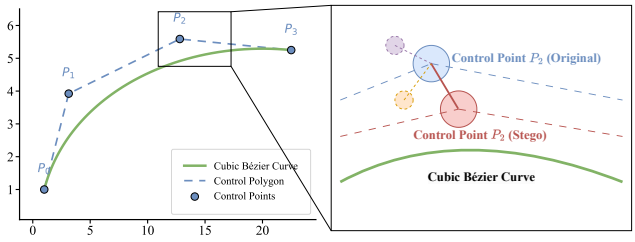


Figure 5. Visual example of encoding information into control points via perturbation.

more, since SVG is stored as plain text, modifications can be made precisely, laying the foundation for our method. As shown in Figure 5, we perform steganographic embedding by defining a mapping function $\mathcal{S}(\cdot, m)$ between the secret information and the perturbation of the control points of the cubic Bézier curve. This process can be expressed as follows:

$$\mathrm{SVG}_{\mathrm{Stego}} = \mathcal{S}\left(\mathcal{F}_{\mathrm{vec}}\left(\mathcal{G}(\theta)\right), m\right), \tag{12}$$

where $\mathcal{G}(\theta)$ denotes the image generated by an image generation model conditioned on the settings θ (e.g., seed and prompt), $\mathcal{F}_{\mathrm{vec}}(\cdot)$ represents the deterministic vectorization function that converts the generated image into a vector image, m is the secret message to be embedded, and $\mathrm{SVG}_{\mathrm{Stego}}$ denotes the final SVG file containing the hidden information.

Specifically, $\mathcal{S}(\cdot, m)$ represents a control point perturbation strategy based on information mapping. This operation enables efficient information embedding by quantizing and encoding two key parameters: the perturbation length and perturbation direction of the curve control points. First, in the perturbation length dimension, we divide the interval $[0, L_{\mathrm{Max}}]$ into subintervals of uniform length, each corresponding to a unique message. L_{Max} denotes the maximum perturbation distance that can be applied to a control point while keeping the resulting MSE negligibly small and preserving the visual appearance and structure of the curve. Second, in the perturbation direction dimension, we partition the angular range $[0°, 360°)$ into uniform subintervals. In this way, the secret message is mapped to a specific combination of perturbation length and perturbation angle, thereby enabling information hiding.

We introduce the notion of the invertible point. Specifically, assuming that the SVG coordinates are represented with at most k decimal places, we restrict our processing to control points whose coordinates are specified with exactly k decimal places. Next, we search within the quantized coordinate space for candidate points that satisfy the grid constraint at a precision of k decimal places. The resulting perturbed point is then inversely decoded to determine whether it can still be mapped back to the secret message. Only those candidate points whose decoding results remain identical to the secret message are designated as invertible points. In this way, invertible points ensure consistent and reliable embedding and extraction.

For the receiver, the original image can be regenerated using the pre-shared settings. After vectorization, the regenerated vector image is compared with the received stego vector image to extract the hidden information by computing the differences between their control points.

5. Security Analysis

Given the scarcity of dedicated steganalysis methods for vector images, we present a theoretical security analysis of GVIS in comparison with representative existing meth-

Figure 6. Visual examples of stego vector images generated by GVIS. The first and third rows show original raster images generated by the diffusion model on the CelebA-HQ and LSUN-Bedrooms datasets, respectively, while the second and fourth rows display the corresponding stego vector images.

ods. Existing methods often introduce explicit modifications to file content, which may increase their detectability. For example, [14] essentially segments curves, which significantly increases file size. An attacker may also determine whether a vector image is steganographic by examining whether the curves in the image have been unnaturally segmented. However, our method first generates an image and then performs vectorized steganography. Without the image-generation seed, an attacker cannot accurately reconstruct the original image, making it impossible to extract information through differential analysis. Furthermore, because we use uniformly distributed binary messages to perturb the control points of cubic Bézier curves, the magnitudes and directions of these control-point perturbations are random and remain within a very small range, making them difficult to detect through statistical analysis. In addition, we explicitly account for possible overlap-related risks and avoid such locations during embedding.

6. Experiments

6.1. Experimental Setup

All experiments are conducted in a cloud server environment equipped with an NVIDIA RTX 3090 GPU. All deep learning tasks in the experiments are implemented using the PyTorch framework. The image generation module uses a latent diffusion model [26], while the vectorization process utilizes VTracer [29] for raster-to-vector conversion. In our experiments, the core configuration of VTracer is set to color mode, stacked hierarchy and spline output.

6.2. Performance of GVIS

We compare our method with existing pixel-based and vector-based steganography techniques and evaluate performance using SSIM [30], PSNR [8], bit accuracy, and bit capacity, as summarized in Table 1. Since vector images are inherently different from raster images, we assess image quality by first rasterizing both the original vector images and the stego vector images, and then comparing the two converted raster images. In our comparative experiments, all images are generated unconditionally by latent diffusion models with weights pre-trained on the CelebA-HQ [12] and LSUN-Bedrooms [38] datasets. All methods employ images with a resolution of 256×256 for the experiments. Compared with pixel-based methods, vector-based methods generally achieve higher steganographic capacity and extremely high accuracy at the same image size, owing to the intrinsic properties of vector images. However, our method achieves high embedding capacity without changing the file size or altering the statistical characteristics of the image. Because our method first generates and then vectorizes raster images, it offers a high degree of controllability; as long as the sender and receiver share the same conditions (such as prompts, seeds, and generative models),

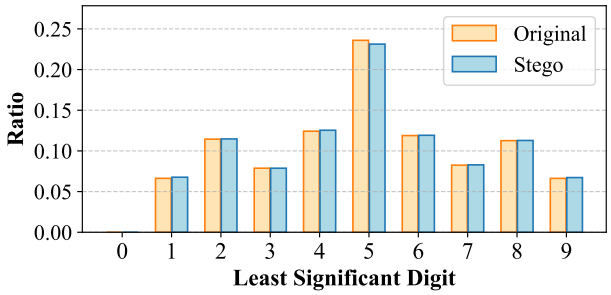


Figure 7. Comparative distribution of least significant digits of control points between stego and original vector images based on the CelebA-HQ dataset.

reliable information transmission is ensured. Several visual examples are shown in Figure 6.

6.3. Security of GVIS

Since dedicated steganalysis methods for vector images are currently unavailable, we first rasterize the stego vector images and then perform steganalysis. Specifically, we evaluate our method using five representative steganalysis networks, namely SiaStegNet [37], StegNet [6], ZhuNet [41], YeNet [36], and XuNet [33]. The results show that the detection accuracy of each method remains around 50%, indicating that the distribution of the stego images is nearly indistinguishable from that of benign images. To further assess the security of GVIS, we analyze the distribution of the least significant digits of its control-point coordinates, with the results presented in Figure 7. These findings indicate that, in contrast to methods that directly manipulate the least significant decimal digit, our method operates at the semantic level and thus preserves the underlying statistical distribution. We also analyze the average file size, as reported in Table 3, where "–" denotes the original SVG files. Our method maintains a file size comparable to that of the original SVG, as it only modifies control points whose coordinates use exactly k decimal places. Since the coordinate precision remains unchanged, the overall file size stays essentially the same.

6.4. Flexibility of Steganographic Capacity

Although the steganographic capacity of GVIS is inherently influenced by generation conditions such as the size of the generated raster image and the vectorization method adopted, once the generation process is fixed, the more important issue is the trade-off among control point perturbation, embedding capacity, and extraction accuracy. In particular, when the angular and distance-based partitions become finer, higher numerical precision is required. However, the decimal precision supported by SVG files limits such fine-grained modifications, which affects the selection of invertible points and may consequently reduce the accuracy of message extraction. To evaluate this effect, we conduct experiments on the CelebA-HQ dataset and consider different partition granularities of the perturbation space according to the bit depth, where n bits correspond to 2^n partitions, and report the resulting average capacity and average accuracy in Table 2. Since our earlier analysis shows that the perturbation radius of control points affects the visual appearance of the curve, we map only 4 bits to the perturbation-length dimension over the range from 0.00001 to 0.01024, corresponding to at most 16 partitions. The experimental results show that, under an SVG precision of eight decimal places, a single control point can embed at least 24 bits while still achieving 100% extraction accuracy. Moreover, even when the embedding capacity is further increased, the extraction accuracy remains high, indicating that the additional capacity can be used to incorporate error-correction codes and thereby further improve extraction reliability.

Table 2. A comparative analysis of capacity and extraction accuracy under varying numbers of interval partitions for control point perturbation length and angle.

Angle Partition	Length Partition	Capacity (bits)	Accuracy
2^4	2^1	18516	1.0000
2^8	2^2	37033	1.0000
2^{12}	2^3	55549	1.0000
2^{16}	2^4	74065	1.0000
2^{20}	2^4	88878	1.0000
2^{24}	2^4	103691	0.9999
2^{28}	2^4	118504	0.9750

Table 3. A comparative analysis of SVG file sizes.

Dataset	Method	Size
CelebA-HQ	-	520.79 KB
	StegoSVG [3]	1485.24 KB
	StegoBIT [13]	799.21 KB
	svgsteg [11]	521.77 KB
	GVIS (Ours)	519.99 KB

7. Conclusion

We propose GVIS, a generative vector image steganography framework. By combining deterministic image regeneration with the precision of vector representations, GVIS offers strong security, controllability, and high embedding capacity. To the best of our knowledge, this study is among the first to explore generative methods for vector image steganography. In the future, we will consider applying this framework to dynamic vector images.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 62472032, and the Young Elite Scientists Sponsorship Program by CAST under Grant 2023QNRC001.

References

[1] Ali Al-Haj. Combined dwt-dct digital image watermarking. *Journal of computer science*, 3(9):740–746, 2007. 1, 2

[2] Evgenia Alexandrovna Blinova. *Steganographic methods and algorithms for copyright protection and ensuring the integrity of electronic documents based on markup languages*. PhD thesis, 2024. 2

[3] Evgeniya Aleksandrovna Blinova and Pavel Pavlovich Urbanovich. Steganographic method based on hidden messages embedding into bezier curves of svg images. *Journal of the Belarusian State University. Mathematics and Informatics*, (3):68–83, 2021. 6, 8

[4] Tu Bui, Shruti Agarwal, Ning Yu, and John Collomosse. Rosteals: Robust steganography using autoencoder latent space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 933–942, 2023. 6

[5] Mehmet Utku Celik, Gaurav Sharma, A Murat Tekalp, and Eli Saber. Lossless generalized-lsb data embedding. *IEEE transactions on image processing*, 14(2):253–266, 2005. 1, 2

[6] Xiaoqing Deng, Bolin Chen, Weiqi Luo, and Da Luo. Fast and effective global covariance pooling network for image steganalysis. In *Proceedings of the ACM workshop on information hiding and multimedia security*, pages 230–234, 2019. 8

[7] Kevin Frans, Lisa Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *Advances in Neural Information Processing Systems*, 35:5207–5218, 2022. 2

[8] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010. 7

[9] Xiaoxiao Hu, Sheng Li, Qichao Ying, Wanli Peng, Xinpeng Zhang, and Zhenxing Qian. Establishing robust generative image steganography via popular stable diffusion. *IEEE Transactions on Information Forensics and Security*, 2024. 2

[10] Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1911–1920, 2023. 2

[11] japplebaum. svgsteg. https://github.com/japplebaum/svgsteg/blob/master/svgsteg.pdf, 2011. PDF document in the GitHub repository "svgsteg: An experiment in steganography", accessed 2026-03-24. 6, 8

[12] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 7

[13] Oleksiy Kinzeryavyy, Iryna Kinzeriava, Alexander Olenyuk, and Krzysztof Sulkowsky. Steganographic method of bitwise information hiding in point-defined curves of vector images. In *International Conference on Computer Science, Engineering and Education Applications*, pages 478–486. Springer, 2018. 6, 8

[14] Alexandr Kuznetsov and Anna Kononchenko. Data hiding in vector images. In *2021 IEEE 16th international conference on computer sciences and information technologies (CSIT)*, pages 171–176. IEEE, 2021. 7

[15] Alexandr Kuznetsov, Anna Kononchenko, and Natalia Kryvinska. Hiding data in vector images: software implementation and experimental research. *Multimedia Tools and Applications*, 82(10):14581–14607, 2023. 2

[16] Jun Li, Ke Niu, Liwei Liao, Lijie Wang, Jia Liu, Yu Lei, and Minqing Zhang. A generative steganography method based on wgan-gp. In *International Conference on Artificial Intelligence and Security*, pages 386–397. Springer, 2020. 1, 2

[17] Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020. 2, 5

[18] Xin Liao, Yingbo Yu, Bin Li, Zhongpeng Li, and Zheng Qin. A new payload partition strategy in color image steganography. *IEEE transactions on circuits and systems for video technology*, 30(3):685–696, 2019. 1, 2

[19] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16314–16323, 2022. 2, 5

[20] Branislav Madoš, Ján Hurtuk, Marek Čopjak, Peter Hamaš, and Michal Ennert. Steganographic algorithm for information hiding using scalable vector graphics images. *Acta Electrotechnica et Informatica*, 14(4):42–45, 2014. 1, 2

[21] KA Navas, Mathews Cheriyan Ajay, M Lekshmi, Tampy S Archana, and M Sasikumar. Dwt-dct-svd based watermarking. In *2008 3rd international conference on communication systems software and middleware and workshops (COMSWARE'08)*, pages 271–274. IEEE, 2008. 1, 2, 6

[22] Nicholas M Patrikalakis and Takashi Maekawa. *Shape interrogation for computer aided design and manufacturing*. Springer, 2002. 4

[23] Yinyin Peng, Donghui Hu, Yaofei Wang, Kejiang Chen, Gang Pei, and Weiming Zhang. Stegaddpm: Generative image steganography based on denoising diffusion probabilistic model. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 7143–7151, 2023. 2

[24] Søren Rasmussen, Karsten Østergaard Noe, Oliver Gyldenberg Hjermitslev, and Henrik Pedersen. Deepmorph: A system for hiding bitstrings in morphable vector drawings. *arXiv preprint arXiv:2011.09783*, 2020. 1, 2

[25] Juan A Rodriguez, Abhay Puri, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images and text. In

Proceedings of the Computer Vision and Pattern Recognition Conference, pages 16175–16186, 2025. 2

[26] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 7

[27] Peter Schaldenbrand, Zhixuan Liu, and Jean Oh. Styleclipdraw: Coupling content and style in text-to-drawing translation. *arXiv preprint arXiv:2202.12362*, 2022. 2

[28] Ron G Van Schyndel, Andrew Z Tirkel, and Charles F Osborne. A digital watermark. In *Proceedings of 1st international conference on image processing*, pages 86–90. IEEE, 1994. 1, 2

[29] VisionCortex. Vtracer. https://github.com/visioncortex/vtracer, 2022. 2, 7

[30] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 7

[31] Ping Wei, Ge Luo, Qi Song, Xinpeng Zhang, Zhenxing Qian, and Sheng Li. Generative steganographic flow. In *2022 IEEE international conference on multimedia and expo (ICME)*, pages 1–6. IEEE, 2022. 1, 2

[32] Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. Svgdreamer: Text guided svg generation with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4546–4555, 2024. 2, 5

[33] Guanshuo Xu, Han-Zhou Wu, and Yun-Qing Shi. Structural design of convolutional neural networks for steganalysis. *IEEE Signal Processing Letters*, 23(5):708–712, 2016. 8

[34] Jianhua Yang, Danyang Ruan, Jiwu Huang, Xiangui Kang, and Yun-Qing Shi. An embedding cost learning framework using gan. *IEEE Transactions on Information Forensics and Security*, 15:839–851, 2019. 1, 2

[35] Zijin Yang, Kejiang Chen, Kai Zeng, Weiming Zhang, and Nenghai Yu. Provably secure robust image steganography. *IEEE Transactions on Multimedia*, 26:5040–5053, 2023. 2

[36] Jian Ye, Jiangqun Ni, and Yang Yi. Deep learning hierarchical representations for image steganalysis. *IEEE Transactions on Information Forensics and Security*, 12(11):2545–2557, 2017. 8

[37] Weike You, Hong Zhang, and Xianfeng Zhao. A siamese cnn for image steganalysis. *IEEE Transactions on Information Forensics and Security*, 16:291–306, 2020. 8

[38] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 7

[39] Jiwen Yu, Xuanyu Zhang, Youmin Xu, and Jian Zhang. Cross: Diffusion model makes controllable, robust and secure image steganography. *Advances in Neural Information Processing Systems*, 36:80730–80743, 2023. 1, 2

[40] Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *arXiv preprint arXiv:1909.01285*, 2019. 6

[41] Ru Zhang, Feng Zhu, Jianyi Liu, and Gongshen Liu. Depth-wise separable convolutions and multi-level pooling for an efficient spatial cnn-based steganalysis. *IEEE Transactions on Information Forensics and Security*, 15:1138–1150, 2019. 8

[42] Zhili Zhou, Yuecheng Su, Jin Li, Keping Yu, QM Jonathan Wu, Zhangjie Fu, and Yunqing Shi. Secret-to-image reversible transformation for generative steganography. *IEEE Transactions on Dependable and Secure Computing*, 20(5): 4118–4134, 2022. 2

[43] Zhili Zhou, Xiaohua Dong, Ruohan Meng, Meimin Wang, Hongyang Yan, Keping Yu, and Kim-Kwang Raymond Choo. Generative steganography via auto-generation of semantic object contours. *IEEE Transactions on Information Forensics and Security*, 18:2751–2765, 2023. 2

[44] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672, 2018. 2