Provably Secure Covert Messaging Using Image-based Diffusion Processes

Luke A. Bauer, Wenxuan Bao, and Vincent Bindschaedler
University of Florida
{lukedrebauer, wenxuanbao, vbindschaedler}@ufl.edu

***Abstract*—We consider the problem of securely and robustly embedding covert messages into an image-based diffusion model's output. The sender and receiver want to exchange the maximum amount of information possible per diffusion sampled image while remaining undetected. The adversary wants to detect that such communication is taking place by identifying those diffusion samples that contain covert messages. To maximize robustness to transformations of the diffusion sample, a strategy is for the sender and the receiver to embed the message in the initial latents. We first show that prior work that attempted this is easily broken because their embedding technique alters the latents' distribution. We then propose a straightforward method to embed covert messages in the initial latent *without* altering the distribution. We prove that our construction achieves indistinguishability to any probabilistic polynomial time adversary. Finally, we discuss and analyze empirically the tradeoffs between embedding capacity, message recovery rates, and robustness. We find that optimizing the inversion method for error correction is crucial for reliability.**

***Index Terms*—steganography, covert messaging, diffusion**

I. Introduction

Encrypted communication allow a sender and a receiver to exchange messages securely. These methods, especially those available nowadays, can resist even the most resourceful adversaries' attempts at recovering the confidential messages, but they do *not* hide the fact that encrypted communication is taking place. Steganography aims to conceal exchanged messages in innocent looking covers, so that adversaries are unaware that communication is taking place. Traditional steganographic methods achieve this by adding distortion to an existing cover, but such distortion is detectable when the cover is public knowledge and can be compared against. Generative models, especially deep generative models, provide a way to avoid distorting an existing cover, by transforming covert messages into seemingly unique content.

We focus on diffusion models [1], [2], [3], [4], which have recently emerged as state-of-the-art generative models for computer vision, due to reportedly outperforming generative adversarial networks (GANs) [5] in output quality [6]. Like GANs, we can view diffusion models as mapping a noise distribution into a desired distribution (e.g., images). However, sampling in diffusion models works by iterative denoising. Starting from a pure noise sample, the model transforms it into a realistic output through consecutive denoising steps.

Consider a scenario where Alice wants to send a message to Bob using a diffusion model's outputs as a covert communication channel. Concretely, Alice runs the diffusion sampling process based on the bits of her message to Bob so that the output (secretly) embeds those bits. This output image is then sent to Bob indirectly, for example by posting it to some public Internet platform or forum. Bob retrieves the image and then inverts the sampling process using the diffusion model to recover Alice's message. Such a method emphasizes security and concealment over anything else, and as such is ideal for exchanging short, but important messages securely.

There are a number of evident challenges to overcome to achieve this vision. The first challenge is to efficiently and invertibly encode bits into diffusion outputs. The second challenge is to ensure that the embedding process is robust to transformations of the output images. Otherwise, any modifications (such as those applied by websites when displaying an image) would destroy the covert message.

Furthermore, arguably the most important challenge is ensuring security, i.e., ensuring that an adversary cannot detect whether an output image contains an embedded message. The prevalence of AI/synthetic generated media online means that identifying an output as model-generated is not by itself a tell that it embeds a covert message. However, subtle changes in diffusion process due to the embedding method could be detectable by an adversary, especially since it is reasonable to assume the adversary has access to the same diffusion model as Alice and Bob.

In this paper, we propose a construction that embeds bits in the initial latents to maximize robustness and is compatible with off-the-shelf latent diffusion models such as StableDiffusion. More importantly, our scheme does not alter the latent distribution. We show how this property grants our method indistinguishability at every step of the process, and thus the embedded message in a diffusion sample is undetectable by any polynomial time adversary.

Our proposed construction has two other desirable features. First, it incorporates a novel error correction technique. This error correction mechanism is built on a specific inversion technique called EDICT [7], which performs inversion using dual latent spaces. We show how to leverage the dual latent spaces to obtain improved message recovery rates. Second, we show that embedding covert bits directly in the initial

This work has been accepted for publication in the IEEE Conference on Secure and Trustworthy Machine Learning (SaTML). The final version will be available on IEEE Xplore.

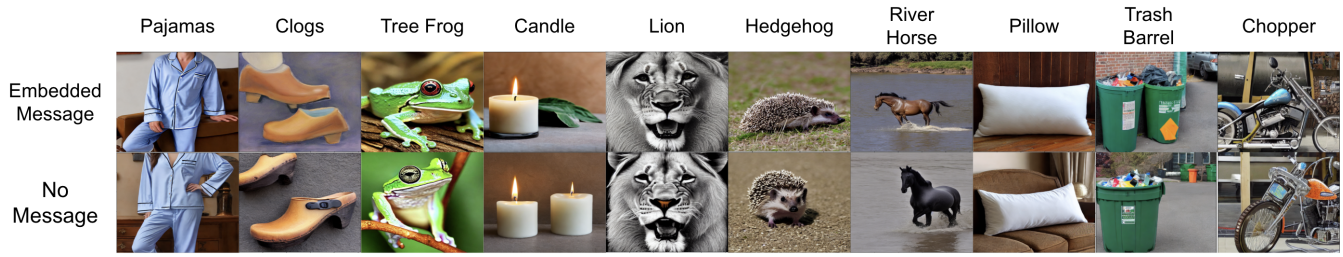


Fig. 1: Sample images generated using our embedding method and the EDICT scheduler. The top of each column is the prompt. The first row of images contain an embedded message, and the second row contain the same original latent space, without an embedded message.

latents noise provides superior robustness to output image transformations (e.g., resize, blur, etc.) than alternatives.

Our paper's contribution is perhaps best conceptualized in relation to the growing literature consisting of numerous prior attempts at doing steganography and (or) covert messaging with generative models such as with text/LLM [8], [9], [10], [11], speech [12], [13], [14], images GANs [15], [16], [17], [18], [19], [20], [21]. However, existing proposals using diffusion models [22], [23], [24], [15] lack robustness, require costly model retraining, and (or) are simply insecure.

Our main argument in this paper is that previous attempts have failed to properly conceptualize security. Instead of considering the various ways that an adversary could attempt to detect if steganographic communication is taking place, most related work simply evaluate the quality of the model under steganographic embedding, concluding that if output quality is high enough then the system is secure. We show that this is a fallacy. Realistic adversaries are *not* limited to examining diffusion outputs. They can use the same inversion process that receivers rely on to recover messages, and use that to identify if steganographic bits are present.

To demonstrate this, we focus on related work Kim et al. [15], the closest prior attempt to ours in terms of embedding strategy. They propose an embedding strategy they call *message projection*. They achieve the high output quality they strive for, but distort the latent noise distribution. We construct an attack that breaks their scheme, despite them achieving the standard steganographic ideal of output indistinguishability.

After describing our construction, we empirically evaluate it and analyze its various tradeoffs, with a focus on capacity, message recovery rates, and robustness. We also show through empirical security evaluation (to complement our security proof) that our scheme is undetectable.

Summary of contributions. We propose a construction with the following properties.

- **Provable security:** we formalize the notion of *latent-space indistinguishability* and prove that our construction achieves it. Unlike related work which only achieves *output indistinguishability* (empirically), latent-space indistinguishability is a stronger notion and under it no adversary can identify that covert messaging is taking place.
- **Distribution-preserving embedding:** we propose a construction that embeds covert messages in the initial latents *without* altering the distribution and prove its indistinguishability.
- **Error correction:** we devise an error correction scheme using EDICT [7]. This improves embedding capacity and increase robustness over the traditional DDIM scheduler.
- **Robustness under image transformations:** we analyze robustness of our scheme to various image transformations (e.g., JPEG compression, resizing, etc.) and show that embedding covert messages in the initial latents provides greater robustness than alternatives.

II. Background & Related Works

A. Diffusion models

Diffusion models [1], [2], [3], [4] can be described through their forward and reverse processes. In the forward process, a clean image $\boldsymbol{x}_0$ is progressively corrupted by adding Gaussian noise through a series of T steps from $\boldsymbol{x}_0$ to $\boldsymbol{x}_T$. The reverse process aims to reconstruct the original image $\boldsymbol{x}_0$ by iteratively denoising the noisy image $\boldsymbol{x}_T$. Starting from $\boldsymbol{x}_T$, the model denoises the image step-by-step, reducing the noise and eventually reconstructing the image at $\boldsymbol{x}_0$.

Forward & Reverse Processes. Among the various sampling techniques used in diffusion models, such as Denoising Diffusion Probabilistic Models (DDPM) [3] and Denoising Diffusion Implicit Models (DDIM) [4], deterministic samplers like DDIM are of particular interest. Deterministic samplers are preferred because they are invertible, allowing the reverse process to be precisely defined and controlled. The invertibility of DDIM is crucial for ensuring that the denoising steps can be accurately traced back, facilitating high-fidelity image generation. The formula of the forward process in DDIM is:

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\epsilon$$

where α_t is a predetermined parameter, and ϵ is a noise vector sampled from the standard normal distribution. The backward process is given by:

$$x_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{x_t - \sqrt{1-\alpha_t}\epsilon_\theta(x_t,t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\epsilon_\theta(x_t,t) + \sigma_t z$$

where $\epsilon_\theta(x_t,t)$ is the noise estimated by the model, intended to reverse the diffusion process, σ_t is an additional noise scale

parameter, and z is random noise sampled from a Gaussian distribution.

In addition to DDIM, we use EDICT [7]. EDICT is a diffusion scheduler that follows a similar structure to DDIM, but creates two latent spaces. These are initially the same space but diverge during the denoising process as they are continually mixed with each other. This uses a similar estimation process that uses mixing to reduce inversion error, especially for models that use complex prompts.

Latent Diffusion Models (LDM). StableDiffusion[2] [25], an advanced application of diffusion models, operates as a Latent Diffusion Model (LDM). LDMs leverage a latent space representation that the diffusion process operates over. The result is that it enhances the model's ability to generate detailed and high-resolution images. The key components of StableDiffusion include not only the diffusion process over a compressed 64x64 latent space but also an encoder and decoder to convert the latent representations back and forth between the full-size images, typically at resolutions such as 512x512. The encoder compresses the high-resolution images into the latent space, which the diffusion process then modifies. Following this, the decoder reconstructs the modified latent image back into a high-resolution output. The stability and efficiency of LDMs make them a preferred choice for complex image-generation tasks.

Sampling and Inversion.

In the sampling process, we start with a text prompt t and an initial latent noise sample $\boldsymbol{x}_T$ (isotropic standard Gaussian [3]), which is represented as a 64x64x4 tensor where the last dimension is the number of channels (RGBA). The StableDiffusion sampling process will then iteratively denoise $\boldsymbol{x}_T$ into $\boldsymbol{x}_0$ (same dimension), and then a full resolution 512x512 image will be decoded from $\boldsymbol{x}_0$. Note that this process is completely deterministic once $\boldsymbol{x}_T$ and the prompt are determined. Given a full size resolution 512x512 image (which may or may not have been created through StableDiffusion) the inversion process first uses the encoder to obtain $\tilde{\boldsymbol{x}_0}$ a 64x64x4 tensor representation of the image. It then iteratively noises this representation back into a latent point $\tilde{\boldsymbol{x}_T}$. As mentioned before since the inversion process is only approximate, the inversion of a sampled initial latent will have lost information. In other words, $\boldsymbol{x}_T \neq \tilde{\boldsymbol{x}_T}$ although we expect $\boldsymbol{x}_T \approx \tilde{\boldsymbol{x}_T}$.

B. Generative Steganography

Since the advent of deep learning there has been significant interest in doing steganography using deep generative models such as LLMs for text-based steganography [11], [9], [8], [10], GANs for images [26], [27], [28], and audio/speech [29], [30]. In this method, the receiver extracts the message from the generated "cover media" through an inversion process or a separate model.

[2]https://stability.ai/ & https://huggingface.co/stabilityai/stable-diffusion-2-1

Several researchers have explored image steganography using neural network frameworks such as GAN [16] or Glow [17]. But were limited by image quality constraints.

Diffusion models have surpassed these other models. Jois et al. [31] propose a scheme called "Pulsar" for embedding secret data in diffusion model outputs, however, it is designed for pixel space diffusion models, not the more popular latent space models.

There are several schemes using latent space diffusion models, but their approaches vary. Peng et al. [22] developed "StegaDDPM", embedding secret messages via denoising diffusion models. But this method suffers from minor distortions leading to inversion failures. Yu et al. [23] propose a method called "CRoSS", a steganography framework using DDIM to convert cover to secret images, differing from traditional embedding methods. Wei et al. [24] propose Generative Steganography Diffusion, using an invertible model and DCT for data embedding. However, their approach requires significant resources to retrain the model. Peng et al. [32] proposed "LDStega," a robust steganography technique for embedding confidential data within images. They demonstrate its resilience to various image distortions, including Gaussian noise and compression. Concurrent work by Mahfuz et al. [33] is PSyDUCK. This method embeds a message by diverging the generation process at certain timesteps in the denoising process; it then mixes the diverged samples back together using the covert message. Observing the final image, the receiver can identify the message used to mix the divergent paths. They apply this method to pixel-based and latent space diffusion models for images and videos.

In contrast to existing techniques, our method both uses off-the-shelf latent diffusion models without retraining, and it embeds covert messages in the initial latents ("latent noise") rather than later on in the diffusion process to obtain greater robustness to modification of the output (e.g., resize, compression, etc.).

Given these desiderata, the work most closely related to ours is Kim et al. [15]. They introduce three distinct message projection methods, each designed to embed secret messages into latent noise. The major limitation of their technique is that it alters the latent space distribution. This may not seem to be a significant issue as their outputs appear realistic. But in fact (as we show empirically) it leads to a total break. An adversary can efficiently detect their cover messages with probability almost 1. By contrast, our proposed technique exactly preserves the latent space distribution and we prove it achieves indistinguishability.

Table I provides an overview of related work in comparison to our approach. In particular, our paper is the only construction that achieves provable security and proposes an error correction scheme.

C. Watermarking

A different problem that relies on similar techniques to ours is watermarking; in which a model owner embeds a small watermark within the generated diffusion output. This problem

TABLE I: Comparison of our approach with related works.

Paper	Retraining Not Required	Robust to Transformations	Embedding in Initial Latents	Preserves Latent Distribution	Error Correction	Provable Security
Peng et al.[22]	✓	×	×	✓	×	×
Yu et al.[23]	✓	✓	×	×	×	×
Wei et al.[24]	×	×	✓	×	×	×
Peng et al. [32]	✓	✓	×	×	×	×
Kim et al. [15]	✓	×	✓	×	×	×
This Paper	✓	✓	✓	✓	✓	✓

has received substantial attention recently [34], [35], [36], [37]. In particular, two methods also attempt to invisibly embed information in the latent space. Tree Ring Watermarking [34] embeds in patterns structured in Fourier space that are highly robust, but influence model output. Gaussian Shading [35] seeks to match the natural Gaussian distribution, and achieves high robustness, but has limited capacity. We discuss these papers because watermarking has similarities to generative steganography in that both attempt to embed information into generated images without decreasing the quality of the output. However, they differ in their goals and threat models. Watermarking strives for easy retrieval and robustness against alterations. By contrast, the goal of steganography is to create a reliable channel between the sender and receiver that is completely undetectable and maximizes its capacity.

III. Problem Statement & Threat Model

We consider a sender and a receiver communicating covertly by using diffusion-generated images. The sender, Alice, embeds a plaintext message into the generation process of a (public) diffusion model. This process results in images that are then sent to Bob, the intended receiver. Bob then proceeds to invert the diffusion process to recover the message.

In this paper, we are not concerned with how the diffusion output images are sent to Bob. For example, they may be shared over a public platform such as Reddit or X. Or in some cases Alice and Bob may in fact be the same individual and the covert messaging channel created is thus a kind of plausibly deniable storage scheme that Alice can use (e.g., Alice has to surrender her personal computing devices at a border crossing to/from an authoritarian regime).

Regardless of specifics, the scenarios we consider prioritize security and reliability over transmission capacity. Therefore, the construction we propose is not designed for frequent low-latency large capacity communication, such as private messaging, but is intended for covert communication (of small payloads) that must not be discovered.

To ensure confidentiality, integrity, and authentication, the message embedded by Alice will first be encrypted and authenticated using a shared key with Bob. We assume Alice and Bob have already shared a cryptographic key pair before initiating any communication.

A. Threat Model

We assume an adversary whose goal is to identify that covert communication is taking place and (if so) who are the communicating senders and receivers. The adversary's goal is therefore to distinguish between diffusion-generated images that contain covert messages and those that do not.

The adversary has access to every aspect of our proposed system, except for the shared secret key between the sender and the receiver. This means the adversary has full knowledge of/access to the diffusion model, hyperparameters used, details of the cryptographic algorithms used, prompts, etc. In particular the adversary can generate their own outputs from the diffusion model, as well as attempt to invert any image back into (reconstructed) latents. However, the adversary does not observe the initial randomness chosen in the sender's latents. We also assume that the adversary runs in probabilistic polynomial time (PPT) in the security parameter (in our case the size in bits of the shared key materials). This is because otherwise the adversary could break the cryptography to recover the plaintext directly (ignoring how the diffusion-based generation works).

We emphasize that the adversary can invert an image using the diffusion model, retrieving the same latent space as the receiver. In other words, the adversary can act as a sender/receiver attempting to decode messages as well as sending fake ones.

Moreover, the adversary can train a (PPT) distinguisher and query the model to identify its distribution, as well as the distribution of the noise used in the image generation process.

We consider the possibility that diffusion-generated images will be altered by image transformations such as resizing or format transformations (e.g., by an adversary or by the platform). This is mostly to reflect the fact many platforms perform such transformations when attempting to display images in certain sizes or shapes, such as for a user portrait or when compressing the image to reduce load on the site. This is not meant to capture an adversary that seeks to completely prevent covert messages, as such an adversary could use image transformation that would completely destroy any communication (e.g., resizing the image to 1x1 pixel, replacing it with a black square, etc.). We also do not consider an adversary that seeks to identify steganographic posts based on the subjects of the post, or the behavior of the user.

We do not consider the cases where the adversary controls the process by which the sender and receiver exchange images (e.g, full control of the platform images are posted on). In which case the adversary could perform other attacks, such as deleting images, replacing images with adversarial examples, analyzing posting habits, and observing scraping activity. We leave such analysis to future work.

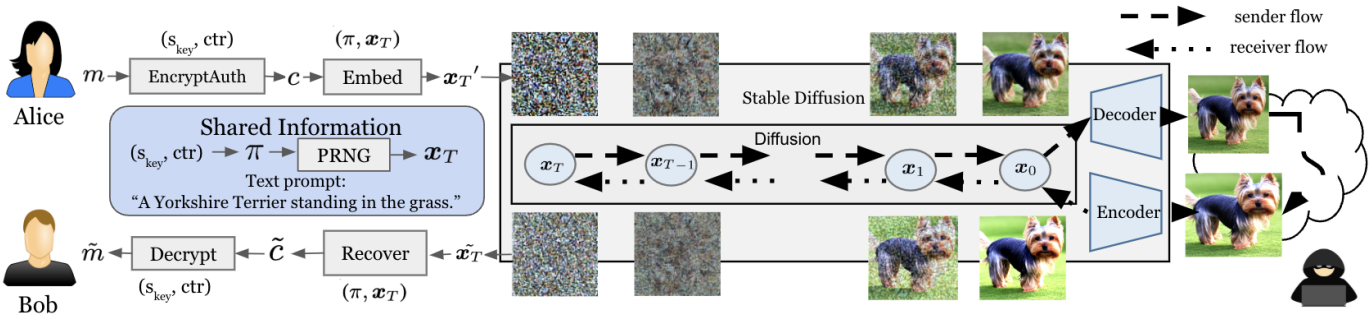


Fig. 2: Proposed construction sender-side and receiver-side operations. Shared information must be agreed upon by both the sender and receiver, but much of it can be stored in the parameter table. Alice encrypts then embeds the message into the latent space which is used to generate the cover image. The cover is then received by Bob, the ciphertext is recovered and then decrypted giving Bob the original message.

B. Security Desiderata & Other Goals

Within the scenario described above there are several goals we wish to achieve.

- **Covertness:** Exchange messages without the adversary being aware an exchange is taking place.
- **Output Indistinguishability:** It should not be possible to distinguish a steganographic generated image from a non-steganographic generated image, without the secret key.
- **Latent Space Indistinguishability:** If the adversary were to attempt to invert a non-steganographic image and a steganographic one, the returned latent spaces should follow the same distribution and be indistinguishable.
- **Confidentiality and Integrity:** The only person who should be able to identify that a covert message is embedded within the noise should be Bob, who uses his secret key to retrieve the message. Furthermore, Bob should be able to determine if the message was altered.

In addition, the steganographic method should be useful for covert messaging. This requires adding few goals relating to performance.

- **Utility:** Alice must be able to embed enough bits per diffusion output for transmitting useful messages, and Bob must be able to reliably extract the messages (with reasonably high probability of correctness).
- **Robustness:** The embedding method should be robust enough to retrieve the message after enduring transformations common on public sites, such as quality reduction, compression, and format changes.

If the adversary alters the images Bob may not be able to retrieve them. However, this should not result in the adversary being able to identify a steganographic image, nor should it result in an image Bob could mistake for the original.

IV. Proposed Construction

Our method can be described in four parts: (1) setup; (2) cryptographic record; (3) embedding; and (4) retrieval. Fig. 2 provides an overview of our construction.

A. Initial Setup

In generative steganography, the sender and receiver must use the same model with the same prompt and parameters, as well as the same steganographic settings. Otherwise they will observe different distributions and be unable to exchange information. We assume the sender and receiver have exchanged a secret key s_{key}, which the adversary does not observe, as well as a message counter ctr for the number of messages exchanged using this key. From this they can then derive unique key materials for each message from this key and message counter. We use this key material to seed PRNGs that are used to generate the initial latent $\boldsymbol{x}_T$, permutation π, and select a row in the parameters table.

The parameter table associates sets of parameters together so that the sender and receiver can stay synchronized in their use of parameters. Each row in the table includes information such as the model name/version, the prompt, the threshold and redundancy levels, and any other information needed to synchronize states. This allows the sender and receiver to exchange multiple messages with varying parameters with only a one time exchange of the secret key s_{key}. The parameter table is ideally private, but that may not be realistic, for logistical and adversarial reasons, so we can assume the table is public. As such the sender and receiver could download it from a public location, ensuring they have the same model and parameter settings. Consistent with our threat model, this means the adversary also has access to this table, and has knowledge of all models and parameters. We assume the models used by the sender and receiver are versions publicly available on sites that host models/finetunings of image generation models. This is not a problem, because security relies only on the secrecy of the key.

It is worth emphasizing that we need not assume anything about the model or its training set. Moreover, if the adversary were somehow able to corrupt the model so that the sender and receiver downloaded different versions, that could prevent correct exchange of messages, but would not undermine the security of the scheme. If this is a concern the sender and

Algorithm 1 Embed Ciphertext into Latent Space

Input: c: Ciphertext;
π: Permutation;
$\boldsymbol{x}_T$: Initial latents;
ρ: Redundancy parameter;
τ: Threshold parameter;
Output: $\boldsymbol{x}_T{}'$: Latents with embedded ciphertext c

1: $c' \leftarrow \text{AddRedundancy}(c, \rho)$ ▷ Within c convert all 0 bits to -1, then apply redundancy by repeating each bit ρ times.
2: $\text{c}'_\text{m} \leftarrow \text{Permute}(c', \pi)$ ▷ Permute c', according to π
3: $\boldsymbol{x}_T{}' \leftarrow \boldsymbol{x}_T$
4: **for** $x_{T,i}$ in $\boldsymbol{x}_T{}'$ **do** ▷ Loop over each latent component in $\boldsymbol{x}_T{}'$
5: **if** $|x_{T,i}| \geq \tau$ **then** ▷ Only embed in values larger magnitude than τ
6: $s \leftarrow c'_m.\text{pop}()$
7: **if** $s = -1$ **then**
8: $x_{T,i} \leftarrow -x_{T,i}$ ▷ Flip the sign of ith element of $\boldsymbol{x}_T{}'$
9: **end if**
10: **end if**
11: **end for**
12: **return** $\boldsymbol{x}_T{}'$

receiver can use hash/checksums to ensure integrity of their local copy of the model.

For simplicity of exposition, we sometimes omit embedding parameters or hyperparameters as input to the algorithms. But we assume both sender and receiver have agreed upon a set of values to use.

B. Cryptographic Record

Before Alice can send the plaintext message m to Bob, she first must encrypt it and ensure integrity and authenticity.

For this, we use an Authenticated Encryption with Associated Data (AEAD) construction realized in practice using an AES block-cipher in CTR-mode to encrypt our plaintext giving us c_{body}. The counter allows them to exchange multiple messages using the same key without compromising message security and integrity. We also generate an authentication tag, c_{tag}, by computing an SHA512-HMAC of c_{body} using the s_{key} and truncating the result (e.g., to five bytes in experiments). The tag c_{tag} is appended to the end of c_{body} giving us our ciphertext $c = c_{\text{body}}||c_{\text{tag}}$. The tag can be checked by the receiver on the retrieved message to ensure no bits were lost from inversion or adversarial interference. Both the AES block cipher and the HMAC tag use keys derived from the secret key s_{key} initially exchanged between the sender and receiver.

C. Ciphertext Embedding

Recall that due to their shared key material, Alice and Bob have agreed on parameters. For StableDiffusion, the initial latents $\boldsymbol{x}_T$ are represented as 64x64x4 tensors sampled as isotropic Gaussian [3]. Said differently, each latent component $x_{T,i}$ is sampled i.i.d. from $\mathcal{N}(0, 1)$. So the only difference in this case is that the PRNG is seeded based on key material derived from the shared secret key and the message counter. This way Alice and Bob will both obtain the same initial latents $\boldsymbol{x}_T$ for each message and can use a similar process to generate the same permutation π.

Algorithm 2 Recover Ciphertext

Input: $\boldsymbol{x}_T$: Initial latents;
π: Permutation;
$\tilde{\boldsymbol{x}_T}$: Reconstructed latents;
ρ: Redundancy parameter;
τ: Threshold parameter;
$p(\cdot)$: pdf of $\mathcal{N}(0, 1)$; ▷ Assume errors are standard normal
Output: $\tilde{c}$: reconstructed ciphertext

procedure FLATTEN($\boldsymbol{x}_T, \tau$)
We wish to convert a 64x64x4 latent space into a single flat list of valid components.
$v \leftarrow []$
Going layer by layer, append to v each latent space component from $\boldsymbol{x}_T$ if their absolute value is greater than τ.
return v
end procedure
procedure REMOVE_MASK($\text{v}, \pi, \rho, \tau$)
$\text{v}' \leftarrow \text{Inverse_permute}(\text{v}, \pi, \tau)$ ▷ Given a list of latent space values remove the applied Mask by identifying all values with a magnitude greater than τ, and then inverting the permutation. Returning latent space values respective to c' (c with redundancy applied)
$\text{v}' \leftarrow \text{Partition}(\text{v}, \rho)$ ▷ Partition list into a list of ρ length lists such that sub-list i contains all the latent space values embedded with the ith bit of c
return v'
end procedure
$\text{v}_{\text{orig}} \leftarrow \text{Flatten}(\boldsymbol{x}_T)$
$\text{v}_{\text{recov}} \leftarrow \text{Flatten}(\tilde{\boldsymbol{x}_T})$
$\text{v}'_{\text{orig}} \leftarrow \text{Remove_Mask}(\text{v}_{\text{orig}}, \pi, \rho, \tau)$
$\text{v}'_{\text{recov}} \leftarrow \text{Remove_Mask}(\text{v}_{\text{recov}}, \pi, \rho, \tau)$
$\tilde{c} \leftarrow$ “”
for $i \leftarrow 0$ to len(c) **do**
Use Eq. (1) to recover the ith bit
Add the bit to $\tilde{c}$
end for
return $\tilde{c}$

Given this, Alice has to perform two steps for embedding a ciphertext $c = c_{\text{body}}||c_{\text{tag}}$. First, redundancy is added to the ciphertext as follows. Each bit in c is repeated $\rho \geq 1$ times (redundancy parameter) before permuting the (redundant) ciphertext according to permutation π, creating c'_m. Second, we only embed bits into component values (of $\boldsymbol{x}_T$) with absolute values above a certain threshold $\tau \geq 0$. So for each component value within the 64x64x4 latent space, if its absolute value exceeds τ we embed a bit from c'_m. To embed a 1 bit, we do not change the component value, and to embed a 0 bit, we flip the current sign of the component value. As we show in Section V-D, this flipping of bits does not alter the distribution. The method is illustrated in Algorithm 1. Lines 7-8 perform the actual logic embedding bits. The process can be thought of as applying a mask of permuted values over valid values. If we have redundancy ρ then we can only embed $16384/\rho$ bits of the ciphertext at the most per output (assuming 64x64x4 latent space), and any component value that does not exceed τ in magnitude is not embedded with a bit. We discuss this further in Section VIII.

D. Latent Space Inversion

Our construction uses the DDIM scheduler inversion process proposed by Ho et al. [3] and as such can be used with

Algorithm 3 Error Correction

Input: s_{key}: secret key;
$\tilde{c_1}$: reconstructed ciphertext of first latents;
$\tilde{c_2}$: reconstructed ciphertext of second latents;
max_errs: maximum potential errors to correct ▷ default: 10
Output: m: message, or Fail

```
procedure CHECK(c, s_key)
    Check tag of ciphertext, if valid return True, else return False
end procedure
err ← Compare(c̃_1, c̃_2)  ▷ Identify indexes where the ciphertexts differ.
if len(err) > max_errs then
    return Fail
end if
Use recursion to generate 2^max_errs possible ciphertexts.
Each ciphertext being a possible combination of error bit replacements.
for c in 2^max_errs possible ciphertexts do
    if Check(c, s_key) then
        return Decryption of valid c
    end if
end for
return Fail
```

any diffusion model using that scheduler. In this paper, in addition to using the DDIM scheduler, we also propose to use EDICT, a variant of DDIM that uses dual latent spaces to improve inversion accuracy [7]. More specifically, when denoising at each step, the scheduler will update one latent space with information from the other one, in a deterministic and invertible way. This helps stabilize the inversion process, allowing us to perform error correction. The use of EDICT increases message accuracy (the percentage of messages sent that can be perfectly recovered) as we detail in Section VIII. When embedding, the only difference between EDICT and DDIM is that we duplicate $\boldsymbol{x}_T{}'$, to form $(\boldsymbol{x}_T{}', \boldsymbol{x}_T{}')$ which functions as the dual latent spaces that initiate image generation. On the receiving end we perform the same retrieval process across both latent spaces, and then use them to perform error correction as detailed in Algorithm 3.

E. Ciphertext Recovery

Once an image has gone through inversion, the receiver has an inverted latents $\tilde{\boldsymbol{x}_T}$ (or $(\tilde{\boldsymbol{x}_T}, \tilde{\boldsymbol{x}_T}')$ in the case of EDICT). Recall that the receiver also knows $\boldsymbol{x}_T$, the initial latents used by the sender for this message. Using $\boldsymbol{x}_T$ to identify which component values had magnitudes above the threshold τ, the receiver can compute the mask used by the sender and extract the component values associated with it, and then invert the permutation to recover latent components that encode the ciphertext, with redundancy.

Once the mask is removed, the receiver has a list of component values matching the ciphertext with added redundancy. The receiver then splits the list into $\text{len}(c)$ lists of length ρ. Each list in v'_{recov} contains ρ component values, which were embedded with the redundant bits for bit i in the ciphertext. Thus each list $\text{v}'_{\text{recov}}[i]$ can be associated with the i-th ciphertext bit. We repeat this process on the initial latents $\boldsymbol{x}_T$, to receive v'_{orig}, with the same list format but containing component values from $\boldsymbol{x}_T$.

For each bit i, the receiver then compares $\text{v}'_{\text{orig}}[i]$ and $\text{v}'_{\text{recov}}[i]$. If all components of each of the two lists have the same sign, then recovering the value of embedding bit i is straightforward. It is 1 if the signs in $\text{v}'_{\text{recov}}[i]$ match the signs in $\text{v}'_{\text{orig}}[i]$ and 0 otherwise. (Recall that during embedding Alice flips the sign of each component if the corresponding ciphertext bit is 0.)

The challenge is that because the inversion process is not perfect, within the list $\text{v}'_{\text{recov}}[i]$ there may be components of different signs. In that case, we propose to recover the ciphertext bit using a likelihood ratio test based on the assumption that inversion errors are i.i.d. standard normal distributed. Said differently, we assume that if the value of a component (of $\boldsymbol{x}_T$) is initially x then after embedding and inversion it is $x + z$, where $z \sim \mathcal{N}(0, \sigma^2)$ for some unknown but fixed $\sigma > 0$.

This strategy allows us to take into account the magnitude of each component value in the recovery (as opposed to only its sign). For example, it is more likely that inversion error caused 0.02 to flip to -0.01 than it is that 3.0 flipped to -3.2.

Concretely, the likelihood ratio we use to retrieve the ith bit is:

$$\frac{\prod_{j=0}^{\rho-1} p(\text{v}'_{\text{orig}}[i][j] - \text{v}'_{\text{recov}}[i][j])}{\prod_{j=0}^{\rho-1} p(-\text{v}'_{\text{orig}}[i][j] - \text{v}'_{\text{recov}}[i][j])} \tag{1}$$

where $p(\cdot)$ denotes the standard normal probability density function.[3]

If the ratio in Eq. (1) is greater than 1 then it is more likely (given the standard normal error assumption) that ciphertext bit i was not flipped (so $c_i = 1$). Otherwise, we conclude ciphertext bit i was most likely 0. We use this equation and show the entire process in Algorithm 2.

With this, we can recover the ciphertext. We can then verify that we have correctly retrieved the ciphertext by comparing the tag at the end of the ciphertext against an HMAC of the body of the ciphertext. If they match we have correctly retrieved the message. If they do not then there has been an error. Note that in addition to ensuring integrity, this also proves the message came from Alice since without the key an adversary cannot forge a valid HMAC tag.

When using the DDIM scheduler, this is the end of the recovery process. If the HMAC tag does not match, then the message is lost.

Error Correction with EDICT. By contrast, when using EDICT we can perform additional error correction. Our novel proposal works as follows. Due to EDICT we have dual latents $(\tilde{\boldsymbol{x}_T}, \tilde{\boldsymbol{x}_T}')$ and therefore we can use Algorithm 2 and Eq. (1) to create two reconstructed ciphertexts $\tilde{c_1}, \tilde{c_2}$, one for each inverted latents. Both ciphertext can be checked to see if the HMAC tag is valid. But if neither is recovered without error, we can use their differences to identify and recover correct bits. If the number of differing bits between $\tilde{c_1}$ and $\tilde{c_2}$ is small,

[3]We can use the standard normal PDF for this and do not need to know the common variance σ^2 since it does not affect whether the ratio is greater than 1.

then we can create a new ciphertext for every combination of differing bits and check if the HMAC tag is valid.

In practice, assuming that reasonable embedding parameters ρ and τ were used, there is often only a small number of problem bits (e.g., less than 10) in practice. Therefore we can check every possibility quickly. If we observe n problem bits, then there are 2^n possible ciphertext to check. So, we set a maximum number of problem bits $\mathrm{max_errs}$ to attempt recovery (e.g., $\mathrm{max_errs} = 10$) to bound the computational effort. This method allows recovering any message for which there are only a few errors due to inversion, and for which both retrieved ciphertexts do not incorrectly identify the same bit. If an incorrect bit is not among the problem bits, then the recovered ciphertext is incorrect and the message is lost.

V. Formalizing and Analyzing Security

In this section, we analyze the security of our proposed approach following our threat model, defined in Section III-A.

Defining Security & Indistinguishabilities. Recall that the adversary's proximate goal is to distinguish diffusion images that contain embedded messages from those that do not. Therefore, a straightforward way to define security is as *output indistinguishability*, i.e., the scheme is secure if the output images produced by the diffusion-based steganographic scheme are indistinguishable from images produced by the diffusion model itself. This is an appealing way to define security, in part because it is easy to evaluate empirically. It suffices to generate a dataset of images produced by the steganographic scheme and a dataset of images produced by the diffusion model alone and compare the two, for example, using an ML-based distinguisher. This is what most related works do [22], [23], [24], [15].

The problem with this reasoning is that the adversary is not limited to only analyzing the diffusion images themselves. The adversary can reason about the process by which images are obtained and even simulate the sender or receiver process. Of course, the adversary can never recover the plaintext message embedded in the diffusion image (without the secret key or breaking the cryptography). But by an approximate inversion of the image they can recover approximate latents $\tilde{\boldsymbol{x}_T}'$, which they may use to infer the presence of an embedded message. Therefore, output indistinguishability is insufficient. A secure scheme needs to ensure that the distribution of any recovered latents $\tilde{\boldsymbol{x}_T}'$ on diffusion output embedding some covert messages matches exactly the distribution of recovered latents on (non-steganographic) outputs. A principled way to do this in our setting is to achieve *latent-space indistinguishability*, i.e., ensure that the initial latent-space distributions with and without embedding covert messages are identical.

Latent-space indistinguishability is a stronger notion. It implies output indistinguishability because once initial latents are sampled, the diffusion process is completely deterministic. If the initial latents with and without covert messages are identically distributed, then the output with and without covert messages will also be identically distributed.[4]

In the rest of this section, we prove the security of our construction. First, we formalize initial latents schemes (Section V-A). We then propose an indistinguishability game for them (Section V-B). In the process, we explain why our proposed approach does not suffer from the same issue as Kim et al. [15] (Section V-C). Finally, we prove our construction's indistinguishability (Section V-D).

A. Initial Latents Steganography

We are given a security parameter l_{sec} and a latent space dimension k. A *initial latents steganography* scheme is a tuple (Init, Embed, Recover) defined as follows.

- $\mathrm{Init}(l_{\text{sec}})$ produces a secret key s_{key} of l_{sec} bits.
- $\mathrm{Embed}(s_{\text{key}}, m, \boldsymbol{x}_T)$: encrypts and embeds a message m of $0 < l_m \leq k$ bits under key s_{key} starting from initial latents $\boldsymbol{x}_T \in \mathbb{R}^k$. The output is some $\boldsymbol{x}_T{}' \in \mathbb{R}^k$.
- $\mathrm{Recover}(s_{\text{key}}, \tilde{\boldsymbol{x}_T}, \boldsymbol{x}_T)$: recovers plaintext from inverted latents $\tilde{\boldsymbol{x}_T} \in \mathbb{R}^k$ and initial latents $\boldsymbol{x}_T$ under key s_{key}. The output is a plaintext $\tilde{m}$ of length l_m or $\perp$ (failure).

The correctness condition is that if for any tuple $(s_{\text{key}}, \boldsymbol{x}_T, m)$, if $\mathrm{Embed}(s_{\text{key}}, m, \boldsymbol{x}_T)$ outputs $\boldsymbol{x}_T{}'$, then $\mathrm{Recover}(s_{\text{key}}, \boldsymbol{x}_T{}', \boldsymbol{x}_T)$ outputs m' such that $m = m'$. In other words, assuming zero inversion loss, the message is recovered with probability 1 under any key and initial latents. In practice, since there will be inversion loss, we are never guaranteed to recover any message.

Observe that the construction does not involve the diffusion model at all. This is by design and a consequence of the fact that we embed bits in the initial latents.

B. Indistinguishability Game

We propose an indistinguishability game to formalize latent space indistinguishability for diffusion. This is the main security notion we use in this paper.

We assume that the challenger and adversary have common diffusion model $\mathcal{G}$ with a prompt t and sampling hyperparameters params. We also have l_{sec} the security parameter in bits and a message length $l_m > 0$.

The game proceeds as follows.

1) The challenger generates key material s_{key} of l_{sec} bits and also a uniformly random bit $b \in \{0, 1\}$.
2) The adversary generates a plaintext message m of at most l_m bits and sends it to the challenger.
3) The challenger pads m to l_m bits. The challenger then generates $\boldsymbol{x}_T$ from the diffusion model $\mathcal{G}$ initial latent distribution $\mathcal{D}$. If $b = 1$, the challenger invokes $\mathrm{Embed}(s_{\text{key}}, m, \boldsymbol{x}_T)$ to obtain $\boldsymbol{x}_T{}'$ and then sets $\boldsymbol{x}_T = \boldsymbol{x}_T{}'$. Otherwise (if $b = 0$) then the challenger does nothing.
4) The challenger invokes the diffusion model $\mathcal{G}$ with a prompt t and sampling hyperparameters params on $\boldsymbol{x}_T$.

[4]Once we achieve latent space indistinguishability, we do not need to worry about the output quality of images embedding covert messages; it will be identical to those images that do not embed anything. Nevertheless, we evaluate this empirically in Section VII-B.

It obtains the corresponding image $\boldsymbol{x}_0$ and sends it to the adversary.

5) The adversary then runs PPT (in l_{sec}) algorithm $A(\mathcal{G}, t, \text{params}, \boldsymbol{x}_0)$ to obtain a bit b'. The adversary outputs b'.

If $b' = b$ then the adversary wins the game. Moreover, we define the adversary success rate as $\Pr(b' = b)$ and consequently the adversary's advantage as $\text{Adv} = 2\Pr(b' = b) - 1$.

We say a scheme is secure if no PPT adversary can win the game with advantage greater than a negligible function with respect to the security parameter l_{sec}. In other words, if for every PPT adversary we have: $\text{Adv} \leq \text{negl}(l_{\text{sec}})$.

C. The Problem with Distorting Distributions

Kim et al. [15], our closest prior work, alters the initial latents distribution as part of their embedding process. Indeed, instead of generating a random Gaussian latent space, they use message projection to fill the latent space with values according to the bits they need to embed. So instead of flipping a value's sign (as we propose), they replace it with 1 for one bits, and -1 for zero bits. Alternatively they pull from a small subset of values, i.e. $\sqrt{2}$ or $-\sqrt{2}$ for 1 bits and 0 for zero bits.

The result is that after embedding their latents do not follow an isotropic Gaussian distribution. Their method can match the mean and standard deviation of the standard Gaussian distribution, but in reality there only three possible values (0, $\sqrt{2}$, or $-\sqrt{2}$). This discrepancy in distribution can be exploited by an adversary to identify covert messages.

Under realistic threat models (such as ours) the adversary does not observe the latents directly. Instead they observe diffusion outputs from which they can attempt to reconstruct the latents by inversion. As we show in experiments (Section VII-A), despite significant loss in the inversion process, the proposal of Kim et al. [15] is not secure. An adversary that approximately reconstructs the latent and trains a simple ML-based distinguisher (e.g., a Random Forest classifier) readily achieves 100% accuracy on distinguishing between their latent spaces and non-steganographic latent spaces.

Interestingly, despite altering the latents distribution, the construction of Kim et al. [15] produces high quality diffusion outputs (as we confirmed experimentally — Section VII-B). The pitfall of their approach is the (implicit) assumption that high output quality implies security.

D. Achieving Latents Indistinguishability

Our embedding procedure (Algorithm 1) does not alter the distribution of the initial latents. Intuitively, the reason is that the initial latents are sampled i.i.d. from a symmetric distribution and Algorithm 1 simply flips some of the signs based on the redundancy-expanded ciphertext bits. Interestingly, this argument would not hold true if for example the algorithm was to set the sign based on the redundancy-expanded ciphertext bits (e.g., set $x_{T,i} \leftarrow +|x_{T,i}|$ if bit i is 1 and $x_{T,i} \leftarrow -|x_{T,i}|$ otherwise). For instance, in such a case, the number of positive entries in the transformed latents would be a multiple of the redundancy. This could then be used by an adversary to distinguish latents that embed bits from those that do not.

The following theorem makes this intuition precise. It states that Algorithm 1 does not alter the natural sampling distribution of the underlying (latent) diffusion model.

Theorem 1. *Let $\boldsymbol{x}_T$ be a vector of k random variables each distributed according to a symmetric distribution $\mathcal{D}$, i.e., $x_{T,i} \sim \mathcal{D}$ for $i = 1, 2, \ldots, k$ i.i.d. Also, let* Emb *denote the transformation of $\boldsymbol{x}_T$ into $\boldsymbol{x}_T{}'$ in Algorithm 1. Then the distribution of $\boldsymbol{x}_T$ is not altered by* Emb*, i.e., we have: $\text{Emb}(x_{T,i}) \sim \mathcal{D}$ for $i = 1, 2, \ldots, k$.*

Theorem 1 states that any adversary (even one not bounded in probabilistic polynomial time) cannot distinguish between initial latents $\boldsymbol{x}_T$ and initial latents $\boldsymbol{x}_T{}' = \text{Emb}(\boldsymbol{x}_T)$ that embed ciphertext bits. Since given the latents, the diffusion process is entirely deterministic (and public) the corollary is that diffusion outputs that embed ciphertext bits are also (statistically) indistinguishable from those that do not. It immediately follows that no PPT adversary can win with higher than negligible advantage the indistinguishability game of Section V-B. That said, this does not imply the entire construction is statistically secure (it is only PPT secure). The reason is that a non PPT adversary could break the cryptography itself, even if they cannot directly distinguish the latents that embed ciphertext bits from those that do not.

Proof of Theorem 1. By construction, Algorithm 1 applies to each component of $\boldsymbol{x}_T$ independently (based on each bit in ciphertext c). So it suffices to consider what happens to $x_{T,i}$ for an arbitrary i. Let $\Pr_{\mathcal{D}}(x_{T,i})$ denote the probability density function of $x_{T,i}$ under $\mathcal{D}$. If the embedded bit is 0 then there is no change ($x_{T,i} = \text{Emb}(x_{T,i})$). If the embedded bit is 1 then $\text{Emb}(x_{T,i}) = -x_{T,i}$. But since $\mathcal{D}$ is symmetric, i.e., $\Pr_{\mathcal{D}}(x) = \Pr_{\mathcal{D}}(-x)$ for any $x \in \mathbb{R}$, there is no change in the distribution and we conclude that $\text{Emb}(x_{T,i}) \sim \mathcal{D}$. □

Remark that Theorem 1 only requires that the distribution be symmetric and each component be sampled i.i.d. (e.g., it need not be Gaussian). However, in our case with StableDiffusion the distribution of each latent space component is standard normal distributed (i.e., $\mathcal{D} = \mathcal{N}(0, 1)$).

VI. Experimental Setup

As our LDM, we use CompVis/stable-diffusion-v1-4 [38] with one of two schedulers, DDIM or EDICT. For prompts we use one of ImageNet's 1000 classes [39]. For the diffusion process, we use 50 inference steps, and we generate 512x512 images. Unless otherwise stated we use a threshold of 0.3 and a redundancy of 6, and the EDICT scheduler. All experiments are done on servers with 12 GB of RAM for the CPU, and an NVIDIA A100 NVSWITCH GPU with 80 GB of memory.

VII. Security Experiments

Although we proved the security of our construction, we can nevertheless evaluate security empirically. For this, we can train ML-based distinguishers to identify any distortions

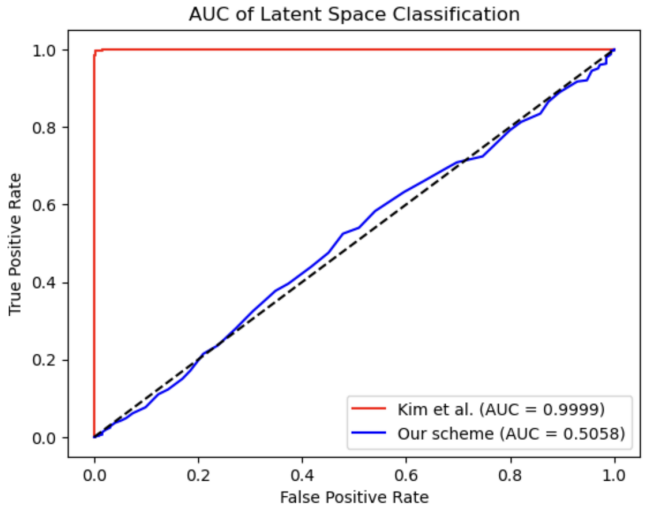


Fig. 3: ROC curves and AUCROC for Random Forest classifiers to distinguish latent space with and without embedding. Our method does not alter the latent space and so classifier performance is random guessing. Kim et al. [15] (message projection) do alter the distribution, which leads to (almost) perfect classification.

caused by the embedding of our messages. We can also evaluate cover (i.e., output) image quality to determine whether embedding covert messages alters the output distribution in a way that an adversary can detect. For this, we use the Fréchet Inception Distance (FID) [40], CLIP-score [41], and image classifiers as distinguishers.

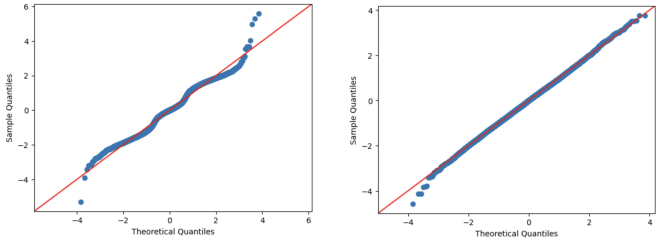
A. Security of the Latent Space

We train Random Forest binary classifiers to distinguish latent spaces. Concretely, we generate 3 sets of 500 latent spaces gathered by inverting a cover image. All images are generated and inverted with a DDIM scheduler. The first set does not contain a steganographic message. The second set contains a message embedded using our method, and the final uses a message embedded by (our re-implementation of) the method used by Kim et al. [15]. Recall that their embedding method overrides the initial latents by assigning 1 bits to either $\sqrt{2}$ or $-\sqrt{2}$ and 0 bits to 0.

We create a histogram for each latent space with 10 equally spaced bins ranging from -3 to 3. The bins are filled with the number of feature values that fall within the range of the bins. We then train a RF classifier on the histogram bins distribution counts as features to distinguish between non-steganographic latent spaces, and ones containing a message embedded using our method. We also trained another RF classifier in the same way to distinguish between non-steganographic latent spaces and message projection. We use 400 training samples and 100 test samples for both classifiers.

The results are shown in Fig. 3. For our method the classifier only achieves an AUC of 0.5, or random guessing. For Kim et al. it achieves an AUC of 1.0, meaning it was able to identify every single steganographic message without errors.

To demonstrate why it is so easy to detect latent space embeddings when they alter the distribution, we also generate



(a) Message Projection (b) Our Method

Fig. 4: QQ plots of the two embedding methods against the natural recovered latent space distribution. Although the graph on the left follows the natural mean and scale well, the clustering around the values inserted into the feature values is easy to detect.

QQ-plots from the same sets of latents. First we establish a 'natural' set, which is the distribution across 500 latent spaces recovered from images without an embedded message. We can then compare the two methods. To create the QQ-plot we create a histogram with 100 equally spaced bins ranging from 3 to -3, and then compare the proportion of points within that bin against the natural distribution. We provide the results in Fig. 4. The results show that simply inserting values according to the bits being embedded leave distinct clusters near the values being used.

To further illustrate the effect of embedding method on returned latent spaces, we performed a two-sample Kolmogorov-Smirnov (KS2) test. We compare the natural set described above against similar sets for our method and for message projection.

Our null hypothesis is that the distributions are the same, and thus the embedding method provides latent space indistinguishability. Using the ks_2samp function of the scipy.stats package, we compare our method against the natural distribution, which returned a test statistic value of 0.0006 and p-value 0.475 and thus we do not reject the null hypothesis. For the message projection method, however, the test statistic is 0.055 with a p-value $4.56 \cdot 10^{-49}$ and hence the null hypothesis can safely be rejected.

B. Security of the cover

We calculate FID using the Pytorch FID implementation [42]. We generate a large set of images. 10,000 without a message embedded, and 10,000 with a message embedded. To establish a base FID, we split the natural dataset in half, then calculated the FID of the two halves. This returned an FID of 13.688. We then calculated the FID of images with a message embedded within them against images without a message, and got an FID of 13.642.

We also gathered the CLIP-scores [41] of the generated images against the prompt used to generate them. Images without a message had an average CLIP-score of 0.363 ± 0.001. Images with a message had an average score of $0.3628 \pm .0008$. Using a t-test with the null hypothesis that these originate from the same distribution, returns a p-value of 0.1199, meaning we do

not reject the null hypothesis, and these are likely the same distribution.

TABLE II: Accuracy of various steganalysis models attempting to detect if a generated image contains an embedded message or not.

	CLIP	Xu-net	Ye-net	SiaStegNet
Model Acc	0.506	0.50	0.497	0.502

C. Steganalysis

We use these same datasets to train several steganalysis models. Clip [43], Xu-net [44], SiaStegNet [45], and YeNet [46]. For CLIP we use OpenAI's CLIP model with ViT-B/32 transformer architecture. We create two classes "steganographic" and "non-steganographic", and attempt to classify images according to them. We use the dataset we created in Section VII-B 10,000 images without a message embedded, and 10,000 with a message embedded. Note, this is different from Clip-Score where we compared the image against the prompt used to generate them. Here the model is used to extract features on for a ML-based distinguisher. Our CLIP model uses an Adam optimizer and measures the cross entropy loss. We train for 30 epochs. The model is unable to distinguish between classes.

Xu-net, SiaStegNet, and YeNet are originally intended to identify the noise present when embedding an image directly into a cover, and thus require an original cover image to compare against the embedded cover. This does not exist for our method, so we use cover images generated with the same latent space as our steganographic cover, but with no message embedded in them. Again, the models are unable to distinguish between non-steganographic and steganographic output. The results for CLIP and Xu-net are in Fig. 5.

We also tested images generated using Kim et al.'s method, and found that none were able to detect messages embedded using their methods. This is consistent with their results, since their method of inserting values maintains enough of the original distribution's shape to maintain image quality.

VIII. Performance

We evaluate and discuss tradeoffs between capacity, reliability and processing time. We emphasize that if we had chosen to embed bits in a way that altered the latent distribution, the tradeoffs would also involve detectability.

A. Metrics

Capacity measures the number of bits that can be communicated in a single cover. Reliability is the measure of how reliable message transmission is; can the sender be sure that their message reaches the receiver?

We measure capacity in bits per pixel, assuming an image size of 64x64 in order to have standardized measures across models and related work. Different models may transform the latent space into different sized images, but since we are embedding the message into the latent space the final image size is largely irrelevant.

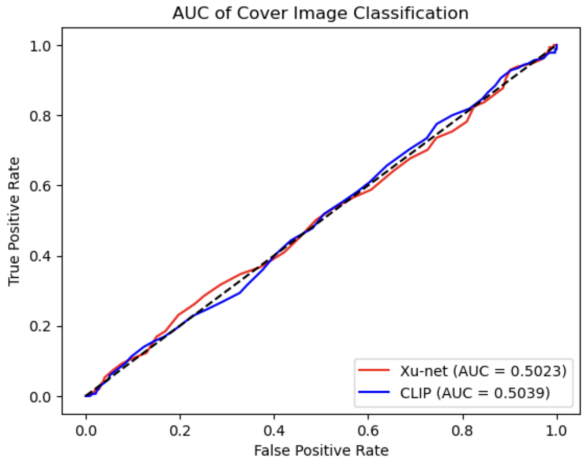


Fig. 5: Cover Image Classification, showing the CLIP and Xu-net models, both unable to distinguish between generated images with a message embedded using our method vs. without. Not shown, these models are also unable to detect images generated by Kim et al.

Reliability can be measured by the likelihood of a message to be received without errors, i.e., it can pass the tag check and be decrypted correctly. To help tune parameters to optimize both of these metrics, we propose a single metric that accounts for both capacity and reliability, called *expected bits received*. This is simply the proportion of sent messages that are received in their entirety, times the number of bits that each message contains. Said differently, it measures the amount of bits correctly recovered per message assuming an infinite stream of transmission from the sender to the receiver. By optimizing for this metric, we can ensure that the maximum number of bits make it to the receiver. We propose this metric because optimizing for capacity or reliability alone is a poor strategy. Achieving high capacity is pointless if the messages sent cannot be recovered correctly. In Table III we show how the metrics of our method compares against similar steganographic and watermarking methods.

A note about bit accuracy. Related work typically evaluates reliability as bit accuracy, i.e., the proportion of embedded bits that the receiver correctly recovers. We use this metric in experiments for ease of comparison, but we do not believe it is meaningful. The reason is that when the message is encrypted (as necessary for confidentiality) retrieving any proportion less than all of the embedded bits prevents recovery of the plaintext, since decryption requires all bits of the ciphertext to be correctly recovered. Therefore, we advocate for using message accuracy, i.e., proportion of complete plaintext messages correctly recovered, which accounts for encryption.

B. Parameter Tradeoffs

The three parameters that affect performance the most are redundancy, threshold, and scheduler. We evaluate redundancy for values between 1 (no-redundancy) and 16, and threshold from 0 to 1.5. For the schedulers we use the standard DDIM scheduler and EDICT. In Table IV we show how the

TABLE III: Metric comparison for similar steganography or watermarking papers. Where multiple methods were present in the paper, we selected the best version. For our results we used metrics for our highest expected value parameters (redundancy $\rho = 6$ and threshold $\tau = 0.3$ and the EDICT scheduler).

(a) Detectability ranges from 0 to 1 with 0.5 being random guessing. Note: For LSB (Least Significant Bit) steganography the capacity is 1 bit per pixel but the embedding is on the final (full size) image, not in the latent space, therefore the actual total capacity is larger.

	Capacity (BPP)	Bit Accuracy	Detectability	Provable Security
Wei et al.[24]	1	1	0.548	×
Peng et al.[32]	0.4	0.995	Unreported	×
Kim et al.[15]	1	0.954	Pe=0.423	×
LSB	1	1	1	×
This Paper	0.501	0.999	0.5	✓

(b) FID and Clip-Scores were measured compared against a natural score without any message embedding. For Tree Ring Watermark, we included the radius of their embedding since they do not provide a capacity measure.

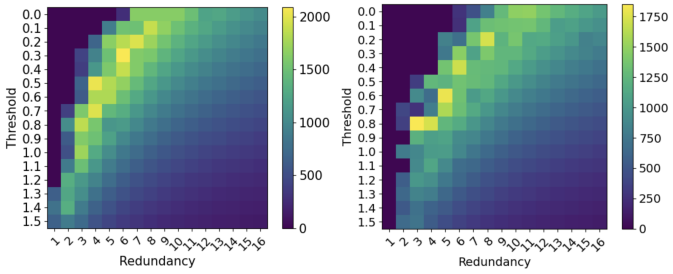
	Capacity (bits)	Bit Accuracy	FID (Base 25.31)	Clip-Score (Base 0.363)
Tree Ring Watermark	r=4	0.9999	25.47	0.356
Gaussian Shading	256	0.9999	25.20	0.3631
This Paper	2087	0.99	25.43	0.3628

TABLE IV: Capacity and reliability for EDICT and DDIM schedulers at different redundancy levels, for a threshold of $\tau = 0.3$. As well as values at different threshold values for a redundancy of $\rho = 6$. Capacity is the same across both schedulers and calculated in bits per pixel (BBP) at the 64x64 size for consistency with prior work and the latent space. Reliability is the percentage of complete messages correctly recovered.

	Threshold ($\tau = 0.3$)					Redundancy ($\rho = 6$)				
	1	2	4	8	16	0	0.1	0.5	1	1.5
Capacity (BPP)	3.056	1.529	0.7639	0.3820	0.1910	0.666	0.613	0.411	0.211	.089
Reliability (EDICT)	0%	0%	7%	93.8%	100%	18.56%	36.4%	91.0%	100%	100%
Reliability (DDIM)	0%	0%	18.4%	86.6%	99.6%	6.4%	20.78%	82.4%	94.5%	100%

redundancy and threshold settings influence reliability and capacity numbers for both schedulers. The capacity does not depend on the scheduler used, only on the latent space size, redundancy, and threshold. However, reliability varies based on the scheduler as different error correction mechanisms are available in each case. Hence we quantify reliability for both. Unless stated otherwise, we use a base threshold of $\tau = 0.3$, and a base redundancy of $\rho = 6$.

Observe that increasing redundancy improves accuracy since our likelihood ratio can better account for inversion inaccuracies, however it decreases the length of messages that can be sent. The threshold has a similar effect, as when we increase it the likelihood of a component value flipping signs due to inversion decreases improving reliability, however fewer component values pass the threshold decreasing the number of slots available to embed bits.



(a) EDICT. (b) DDIM.

Fig. 6: Heatmap of expected bits received at various thresholds (y-axis) and redundancies (x-axis).

What parameters give the highest expected bits received?

We conducted a grid search over threshold and redundancy levels, shown in Figs. 6a and 6b, to identify the optimal parameter values. We tested thresholds from 0 to 1.5 incrementing by 0.1, and redundancy from 1 to 16 incrementing by 1. We generated 10 messages at each combination of threshold and redundancy. We then calculated expected bits received by multiplying the number of message bits contained in a single image at those parameter values by the probability of the message successfully being received.

Interestingly there are diminishing returns for both parameters. The optimal setting is $\rho = 6$ and $\tau = 0.3$, which yields an expected bits received of 2087 (per message).

C. Processing Times

For some deployment scenario, processing time and latency may be important, so we measure it. For our system, excluding image generation and inversion, sender-side operations take approximately 0.3 seconds, and receiver-side operations take 0.5 seconds. Using StableDiffusion as our underlying model with standard DDIM scheduler, takes approximately 20 seconds to generate an image, and another 20 to invert it. Using EDICT doubles these times, so 40 seconds each for image generation and inversion.

Further, recall that EDICT error correction uses the dual latents. So depending on the number of discrepancies, or potential errors, between the latents, error correction could take over 10 minutes. For this reason we only attempt to correct errors if there are less than 10 potential errors, which takes about 0.4 seconds on average.

IX. Robustness

It is common for image hosting platforms to resize, crop, compress, or re-encode images. Therefore, it is desirable to consider robustness, which is the ability to recover the message

TABLE V: Message accuracy at different levels of transformation, SSIM to show image degradation, and average latent difference.

	Resize (Downscale)			Resize (Upscaling)			JPEG Compression			Crop		
	462x462	256x256	51x51	563x563	768x768	972x972	100	75	50	511x511	510x510	509x509
Msg Acc	0.80	0.50	0	0.932	0.834	0.733	0.76	0.40	0.166	0.7	0.365	0.034
Bit Acc	0.999	0.992	0.706	0.999	0.998	0.997	0.999	0.989	0.966	0.998	0.994	0.981
SSIM	0.97	0.86	0.53	0.991	0.989	0.980	0.999	0.962	0.923	0.864	0.92	1.224
Avg Latent Diff	0.34	0.45	0.845	0.302	0.330	0.366	0.311	0.466	0.566	0.368	0.444	0.540

(a) Last Step Embedding (b) Initial Latent Space Embedding

Fig. 7: These are two images that have the same $\boldsymbol{x}_T$ and embed the same message. Left (a) was embedded into the latent space during the last step of the diffusion process. Right (b) follows our original method of embedding into the original latent space.

despite distortions to the image. A common problem with steganographic methods that embed into the cover image directly, is that they are very susceptible to distortions [47]. When one is attempting to store information inside the least significant bits of a pixel, even minor distortions to the pixel values can undermine reliability. Our premise in this work is that by embedding bits in the initial latents, the deleterious effects of distortions will be much less pronounced.

To measure robustness we measured reliability for different image formats, and basic image transformations, commonly applied by social media or image hosting sites. We also measured the Average Latent Difference, defined as the average absolute difference between the latents used to generate the image, and that recovered by the inversion process after the transformation/distortion is applied.

This quantifies how much the transformation affects the latent space, which directly influences our ability to recover the embedded message. To show the effect of the transformation, we also measure SSIM [48] between the image before and after the transformation.

To further illustrate the advantage of embedding information in the initial latents, we compare our method against two alternatives. The first alternative is to embed the message in the last latent step $\boldsymbol{x}_1$ before the latent representation is sent through the encoder to become the full size image. This, theoretically, has the advantage of reducing the inversion error since there is little to no noise added between the embedding of the message and the retrieval. However, this process severely impacts the image quality as can be seen in Fig. 7. The second alternative is to use a method that embeds bits in the image itself, such as Least Significant Bit (LSB) steganography [47].

In Table VI we generate 100 messages for each format, converting the image generated by the model into the format listed, then attempt to retrieve the message from the converted cover. We use the default JPG quality (75), but when increased to 100 or 95, it performs similarly to PNG. We also include results for a blur transformation, which blurs the image by setting each pixel to the average value of the pixels in a square box extending radius pixels in each direction. We use radius 1. We add this as an example of an adversarial attack that could be performed without completely destroying the image.

TABLE VI: Message accuracy, Bit accuracy, average latent difference, SSIM, and accuracy for different embedding methods when switching image formats, and for different transformations.

	Msg Acc	Bit Acc	Avg Latent Diff	SSIM	$\boldsymbol{x}_1$-Msg Acc	LSB Acc
PNG	0.915	0.992	0.386	1	0	100
JPG	0.36	0.807	1.107	0.272	0	0
TIFF	0.865	0.993	0.313	0.96	0	100
BMP	0.885	0.991	0.449	0.94	0	100
Blur	0.533	0.996	0.432	0.86	0	0

Table V shows results for similar tests: 100 samples at the given image transformation level. All distortions are applied using the Python Image Library. JPG is applied by reducing quality to the shown value. For these transformations we use SSIM to establish a range of transformations. We set the minimum to no transformation and the maximum to the point where SSIM stops decreasing, generally when it reaches approximately 0.3. We use this to set 3 levels of transformation. Fig. 8 shows the image transformations.

X. Conclusions & Future Directions

We proposed a novel construction to embed covert messages into the output of image-based diffusion models. In the process, we discussed how embedding in the latent space, without disrupting the natural distribution, provides a secure and useful way to perform steganography.

There remains important avenues for future work. For example, designing an inversion process that further reduces inversion error, would greatly improve usability and robustness. Incorporating ideas from recent research on watermarking could also improve robustness. The security model should be expanded to include user behavior and similarity to other posts within the context. Finally, research into potential misuses of steganographic techniques could be fruitful.

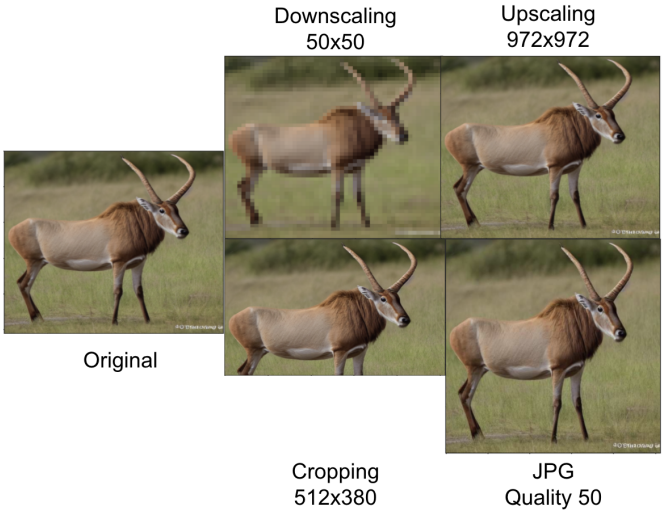


Fig. 8: Sample image transformations at high severity.

ACKNOWLEDGMENTS

We thank the anonymous SaTML reviewers for their helpful comments on earlier versions of the paper. This work was supported in part by the National Science Foundation under CNS-2055123. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[2] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.

[3] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[4] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2020.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[6] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[7] B. Wallace, A. Gokul, and N. Naik, "EDICT: Exact diffusion inversion via coupled transformations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 22 532–22 541.

[8] L. A. Bauer, J. K. Howes IV, S. A. Markelon, V. Bindschaedler, and T. Shrimpton, "Covert message passing over public internet platforms using model-based format-transforming encryption," *arXiv preprint arXiv:2110.07009*, 2021.

[9] G. Kaptchuk, T. M. Jois, M. Green, and A. D. Rubin, "Meteor: Cryptographically secure steganography for realistic distributions," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1529–1548.

[10] Y. Cao, Z. Zhou, C. Chakraborty, M. Wang, Q. J. Wu, X. Sun, and K. Yu, "Generative steganography based on long readable text generation," *IEEE Transactions on Computational Social Systems*, 2022.

[11] F. Z. Dai and Z. Cai, "Towards near-imperceptible steganographic text," *arXiv preprint arXiv:1907.06679*, 2019.

[12] H. Dutta, R. K. Das, S. Nandi, and S. M. Prasanna, "An overview of digital audio steganography," *IETE Technical Review*, vol. 37, no. 6, pp. 632–650, 2020.

[13] J. Li, K. Wang, and X. Jia, "A coverless audio steganography based on generative adversarial networks," *Electronics*, vol. 12, no. 5, p. 1253, 2023.

[14] K. Chen, H. Zhou, H. Zhao, D. Chen, W. Zhang, and N. Yu, "Distribution-preserving steganography based on text-to-speech generative models," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3343–3356, 2021.

[15] D. Kim, C. Shin, J. Choi, D. Jung, and S. Yoon, "Diffusion-stego: Training-free diffusion generative steganography via message projection," *arXiv preprint arXiv:2305.18726*, 2023.

[16] P. Wei, S. Li, X. Zhang, G. Luo, Z. Qian, and Q. Zhou, "Generative steganography network," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 1621–1629.

[17] Z. Zhou, Y. Su, J. Li, K. Yu, Q. J. Wu, Z. Fu, and Y. Shi, "Secret-to-image reversible transformation for generative steganography," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[18] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[19] J. Qin, Y. Luo, X. Xiang, Y. Tan, and H. Huang, "Coverless image steganography: a survey," *IEEE access*, vol. 7, pp. 171 372–171 394, 2019.

[20] M.-m. Liu, M.-q. Zhang, J. Liu, Y.-n. Zhang, and Y. Ke, "Coverless information hiding based on generative adversarial networks," *arXiv preprint arXiv:1712.06951*, 2017.

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

[22] Y. Peng, D. Hu, Y. Wang, K. Chen, G. Pei, and W. Zhang, "Stegaddpm: Generative image steganography based on denoising diffusion probabilistic model," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 7143–7151.

[23] J. Yu, X. Zhang, Y. Xu, and J. Zhang, "Cross: Diffusion model makes controllable, robust and secure image steganography," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[24] P. Wei, Q. Zhou, Z. Wang, Z. Qian, X. Zhang, and S. Li, "Generative steganography diffusion," *arXiv preprint arXiv:2305.03472*, 2023.

[25] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

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

[27] 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. 38 303–38 314, 2018.

[28] J. Liu, Y. Ke, Z. Zhang, Y. Lei, J. Li, M. Zhang, and X. Yang, "Recent advances of image steganography with generative adversarial networks," *IEEE Access*, vol. 8, pp. 60 575–60 597, 2020.

[29] J. Yang, H. Zheng, X. Kang, and Y.-Q. Shi, "Approaching optimal embedding in audio steganography with gan," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 2827–2831.

[30] D. Ye, S. Jiang, and J. Huang, "Heard more than heard: An audio steganography method based on gan," *arXiv preprint arXiv:1907.04986*, 2019.

[31] T. M. Jois, G. Beck, and G. Kaptchuk, "Pulsar: Secure steganography through diffusion models," *Cryptology ePrint Archive*, 2023.

[32] Y. Peng, Y. Wang, D. Hu, K. Chen, X. Rong, and W. Zhang, "Ldstega: Practical and robust generative image steganography based on latent diffusion models," in *ACM Multimedia 2024*.

[33] A. Mahfuz, G. Channing, M. van der Wilk, P. Torr, F. Pizzati, and C. S. de Witt, "Psyduck: Training-free steganography for latent diffusion," *arXiv preprint arXiv:2501.19172*, 2025.

[34] Y. Wen, J. Kirchenbauer, J. Geiping, and T. Goldstein, "Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust," *arXiv preprint arXiv:2305.20030*, 2023.

[35] Z. Yang, K. Zeng, K. Chen, H. Fang, W. Zhang, and N. Yu, "Gaussian shading: Provable performance-lossless image watermarking for diffu-

sion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 12 162–12 171.

[36] Y. Liu, Z. Li, M. Backes, Y. Shen, and Y. Zhang, "Watermarking diffusion model," *arXiv preprint arXiv:2305.12502*, 2023.

[37] Y. Zhao, T. Pang, C. Du, X. Yang, N.-M. Cheung, and M. Lin, "A recipe for watermarking diffusion models," *arXiv preprint arXiv:2303.10137*, 2023.

[38] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 10 684–10 695.

[39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[40] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[41] J. Hessel, A. Holtzman, M. Forbes, R. L. Bras, and Y. Choi, "Clipscore: A reference-free evaluation metric for image captioning," *arXiv preprint arXiv:2104.08718*, 2021.

[42] M. Seitzer, "pytorch-fid: FID Score for PyTorch," https://github.com/mseitzer/pytorch-fid, August 2020, version 0.3.0.

[43] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[44] G. Xu, H.-Z. Wu, and Y.-Q. Shi, "Structural design of convolutional neural networks for steganalysis," *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 708–712, 2016.

[45] W. You, H. Zhang, and X. Zhao, "A siamese cnn for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 291–306, 2020.

[46] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2545–2557, 2017.

[47] F. A. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding-a survey," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1062–1078, 1999.

[48] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.