

SecureImgStego: A Keyed Shuffling-based Deep Learning Model for Secure Image Steganography

Trishna Chakraborty¹, Hasan Murad¹, Imranur Rahman¹, Md. Shohrab Hossain¹, and Shagufta Mehnaz²

¹ Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
{trishnachakraborty13,muradbuetcse13,ir.shimul,shohrab29}@gmail.com

² Dartmouth College, Hanover, NH, USA
Shagufta.Mehnaz@dartmouth.edu

Abstract. Steganography, one of the most popular practices to assure secure transmission of digital message, has numerous applications in modern security. Image steganography is the method of hiding a secret image within a non-secret cover image so that the resulting carrier image dissolves the presence of the secret image. Deploying deep learning-based methods in image steganography has become more prevalent recently. However, such methods are vulnerable to **white-box attacks**. Moreover, **an adversary with access to an image dataset collected from a distribution similar to the training dataset of the original steganography model can train a surrogate model and consequently, retrieve the secret images hidden in the carrier images**. Shuffling the secret images as a pre-processing step can prevent such attacks despite the adversary having access to the original or **surrogate model**. Sophisticated image encryption techniques are not applicable in this scenario since deep learning-based steganography models cannot ensure lossless transmission of the carrier image. Moreover, key concatenation-based techniques proposed for text data steganography are inadequate for image data. To mitigate these issues, in this paper, we have introduced a simple yet effective keyed shuffling approach for the secret images. We have designed keyed pixel shuffling, multi-level block shuffling, and a combination of key concatenation and block shuffling, and embedded them directly within the model architecture to encrypt the secret images. Our findings show that the proposed block shuffling based deep image steganography has negligible error overhead when compared to the vanilla deep image steganography while simultaneously providing strong security against adversary with **white-box access** to the model. Finally, **we perform extensive evaluation of our proposed method and compare it with the existing ones in terms of human perceptibility, key sensitivity, adaptivity, cover image availability, key space, and robustness against steganalysis**.

Keywords: Steganography · Steganalysis · Deep Learning · Block Shuffling.

1 Introduction

In this era of modern technology, assuring privacy and security in computer and internet usage has become one of the most challenging issues. *Steganography* is one of the successful techniques used to secure transmission of secret digital content through unreliable medium. It is the art of hiding a secret message within a non-secret image. Modern steganography offers numerous useful applications such as military communications, securing authorship information, and protection against data alternation. A basic introduction to steganography systems and their applications can be found in [14,15].

Prior research. The traditional approaches for steganography [19,29] utilize different properties of images, e.g., histogram, texture, pixel value difference, and least significant bits. Recently, deep learning-based approaches have become popular among practitioners to design a pipeline for the hiding and revealing processes of steganography [2,33,9]. Baluja [2] proposes the use of convolutional neural networks (CNN) to develop these hiding and revealing processes. The convolutional neural network (CNN) extracts high-level features from the secret image and embeds these features within the cover image. It has been found that deep learning-based steganography technique has significantly high capability of hiding more bits-per-pixel (bpp) secret data within the carrier image compared to the traditional approaches. Generative adversarial networks (GANs) have also been used for designing steganography models [16].

Limitations of existing approaches and challenges. Despite many advanced schemes to embed the properties of the secret message within the carrier image, researchers have shown that the traditional steganography embedding can be identified by leveraging statistical analysis techniques [11,23]. In the case of deep image steganography, an adversary with white-box access to the original model can easily retrieve secret images from carrier images. Moreover, an adversary with access to an image dataset collected from a distribution similar to the training dataset of the original steganography model can train a surrogate of the original model. The adversary can then perform a man-in-the-middle (MITM) attack by retrieving the secret images from public carrier images using the surrogate model’s revealing process. Adversary’s access to the original cover image makes the existing deep image steganography models vulnerable as well. To mitigate such security issues, cryptographic techniques have often been combined with steganography as a pre-processing step [21,26,3,8,10]. Sharma et al. [26] propose a fixed shuffling pre-processing step for image steganography. However, such a pre-processing step cannot support secure steganography for multiple <sender, receiver> pairs due to its nature of fixed shuffling. Moreover, their model is also vulnerable to white-box attacks. Since deep learning-based models for steganography cannot ensure lossless transmission of the carrier image, we cannot utilize advanced image encryption techniques directly in our scenario. Recently, Li et al. [21] have introduced a simple key concatenation-based encryption technique in a deep steganography network for the secure transmission of text data. However, according to our observation, their model is inadequate when the secret message is image data. Therefore, the limitations of the existing

research in terms of applicability in the image domain pose a significant challenge to *secure image steganography*.

Problem and scope. Motivated by the simultaneous requirements of *security* and *utility* of image steganography, in this paper, we address the following research question: *Can we build a deep convolutional neural network (CNN)-based image steganography model capable of incorporating key for each <sender, receiver> pair to ensure security against MITM attackers while also preserving the utility of the model in terms of reconstruction of the secret image?* In response to this, we design and develop **SecureImgStego**, a deep learning-based model for image steganography that embeds keyed shuffling encryption techniques for security while preserving the utility of the model with negligible overhead when compared to the vanilla steganography model [2]. Moreover, we present extensive evaluation and compare different types of key usages along with their trade-offs between security and utility.

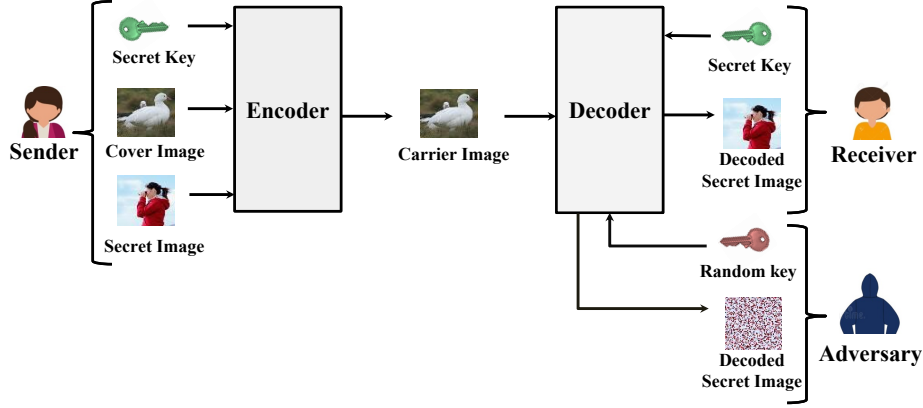


Fig. 1. Sender and receiver share a secret key to encode and decode the secret image, respectively. An adversary with access to the decoder (or, a surrogate model of the decoder) may attempt to decode the carrier image with a random key.

Approach. A high level overview of our proposed image steganography framework is shown in Fig. 1. The framework consists of two main components: an encoder network on the sender side and a decoder network on the receiver side- both designed using convolutional neural networks (CNN). There is a shared secret key between the sender and the receiver. Encoder uses the shared key to encrypt the secret image and then embeds it into the cover image before transmitting to the receiver end. The decoder uses the shared key to decrypt the carrier image and thus recovers the secret image. We capitalize different types of shared keys such as keyed pixel shuffling, multi-level block shuffling, and a combination of key concatenation and block shuffling. To train, validate, and test our model, we utilize *ImageNet*, a public dataset, for both secret and cover images.

Our results show that the keyed block shuffling embedded in the deep learning-based image steganography model architecture introduces negligible error over-

head while providing secure transmission when compared to the existing vanilla deep steganography approaches which are vulnerable to **surrogate model attacks**. Moreover, our proposed framework is robust against both the statistical analysis and steganalysis methods (e.g., StegExpose[4]) unlike the vanilla deep steganography approaches. Finally, our findings show that the flexibility of multi-level block shuffling encryption by varying its depth for different secret images provides optimal performance by **SecureImgStego**.

Contributions. The key contributions of this research work are:

- We design and develop a deep learning-based image steganography model, **SecureImgStego**, for transmitting secret image data by embedding keyed shuffling encryption into the model architecture. **Our model ensures secure communication despite the assumption that the adversary has white-box access to the original model.**
- We demonstrate that the proposed keyed shuffling encryption-based deep image steganography model has negligible error overhead when compared to the vanilla deep image steganography models.
- **We perform extensive evaluation of SecureImgStego and compare it with the existing ones in terms of human perceptibility, key sensitivity, adaptivity, cover image availability, key space, and robustness against steganalysis.**

We organize the rest of the paper as follows. In Section 2, we discuss the related works and identify their limitations. In Section 3, we explain **SecureImgStego** along with the threat model. Section 4 presents the implementation details. In Section 5, we extensively evaluate **SecureImgStego** and report our findings. Finally, we discuss some future research directions and conclude in Section 6.

2 Background and Related Work

In this section, we first introduce the basic terminologies used in the fields of steganography and cryptography. We then discuss the related work along with their limitations and show a gap analysis based on the state-of-the-art research.

2.1 Background

Steganography is a process to conceal confidential data within non-confidential data while also avoiding detection by a man in the middle. These confidential and non-confidential data can be of many types, e.g., text, image, audio, or video. Generally, three data components are associated with this technique: secret object, cover object, and carrier object. A secret object, also known as payload object, refers to the information that needs to be hidden from a man in the middle. A cover object corresponds to an ordinary object where the secret object is embedded. The carrier object is a combination of secret and cover object which is transmitted through the communication channel and thus is publicly available. It has to be visually similar with the cover object to conceal the existence of hidden information in it. Steganalysis, on the other hand, is the process of detecting the existence of (or, recovering) the secret object within (from) a carrier object. In cryptographic systems, keys are used to encrypt information so that any entity without the decryption key cannot decrypt the information.

Broadly, there are two types of cryptography: symmetric and asymmetric. In symmetric cryptography, a shared private key is used for both encryption and decryption processes whereas in asymmetric cryptography, a public key is used for encryption and a private key is used for decryption. In our work, we use symmetric cryptography for secure transmission of the secret object.

2.2 Related Work

To systematize the diversified existing research on steganography, we categorize the related works into three main categories.

Traditional Steganography. Traditional techniques of image steganography utilize different properties of images, e.g., histogram, texture, pixel value difference, or least significant bits [19,29]. The least significant bits (LSBs) replacement is the most popular among the practitioners in image steganography [22]. In this method, the LSBs of the cover image are replaced by the bit pattern of the secret message. It introduces a negligible change in the cover image which is almost imperceptible to the human eyes. Unfortunately, such embedding can be identified by leveraging statistical analysis techniques [11,23].

Deep Steganography. Automatic feature detection capability of deep neural networks has given rise to deep steganography. One eminent work by Baluja [2] proposes an end-to-end deep image steganography framework which consists of three networks: preparation network, hiding network, and reveal network. All bits of the cover image are utilized to hide the secret image in this method which makes it robust against steganalysis. In another image steganography technique, Wengrowski et al. [32] focus on the minimization of camera distortion error by introducing an additional network named Camera display Transfer Function (CDTF) besides encoder and decoder. To hide text data, Tancik et al. [31] encode 56-bit hyperlink bitstring into images. Kreuk et al. [18] present Short-Time Fourier Transform (STFT) and Inverse STFT with vision-oriented models to hide audio data. In the area of compressed image steganography, Sarmah et al. [25] use Cohort Intelligence (CI) algorithm to reduce computational cost.

Encrypted Steganography. There exists a number of research works [6,12,13] that focus on image cryptography. Ding et al. [6] propose a cycle GAN-based approach for medical images where parameters of the network are considered as key whereas Gilad-Bachrach et al. [12] suggest a neural network that is specialized to train on encrypted data. To strengthen data security, various methods use cryptography and steganography simultaneously. A keyed text steganography proposed by Li et al. [20] incorporates key into a model with concatenation operation and demonstrates that even with access to the decoding network, an adversary cannot retrieve the hidden text message from the carrier image if the adversary does not have access to the exact decryption key. In their symmetric-key approach, they show an adversary who elicits secret keys from a uniform random distribution and attempts to decode the carrier image. However, the adversary fails to decode the hidden text message and ends up with random strings. Duan et al. [7] initiate a scheme where image is encrypted at the pre-processing step with Discrete Cosine Transform (DCT) and Elliptic Curve Cryptography

(ECC) algorithms, then fed to the pre-trained neural network, and finally, decrypted with a similar post-processing step. Sharma et al. [28] come up with an idea of incorporating encryption-decryption layer in deep steganography where they use a fixed order of pixel shuffling to encrypt all secret images. With this encryption method, they show that access to the actual cover image is no longer a security threat since the difference between cover and carrier images does not contain the secret image pixels in order.

2.3 Gap Analysis

Existing works on image steganography mainly focus on model architecture improvement, but they fall behind in terms of security issues. An adversary with access to an image dataset from a similar distribution the original steganography model has been trained on, can train surrogates of these models, and therefore, can retrieve the secret images. Unfortunately, the solution proposed in [20] is applicable to only text data and provides no information on how to extend their approach to handle image data. In Subsection 5.3, we show how and why this key concatenation approach fails to extend to image data. Duan et al. [7] have proposed a keyed steganography, but it is a time-intensive and arduous approach with asymmetric ECC and no symmetric keyed solution is mentioned. Moreover, Baluja [2] has shown that if an adversary has access to the actual cover image even without the decoding network, it can partially reveal information about the secret image by enhancing the difference between the cover and carrier images. This limitation has been addressed by Sharma et al. [28] where they hide a scrambled version of the secret image into the cover image. However, since the order of pixel scrambling is constant in their model, an adversary with access to a surrogate model can easily decode the secret image. Besides, their approach is also vulnerable to known-plaintext attacks, i.e., if the adversary can somehow manage to obtain the scrambling order of one secret image, all encoded secret images can easily be retrieved by the adversary using that obtained scrambling order.

The novel contribution of our approach is to establish a simple but fast and effective symmetric keyed shuffling approach so that even if the adversary trains a surrogate model or has the actual cover image, the adversary cannot retrieve the secret image. While designing **SecureImgStego**, we also ensure that the utility of the steganography model is preserved, i.e., we incur negligibly higher error overhead when compared to the insecure vanilla deep steganography model [2].

3 Proposed **SecureImgStego** Model

In this section, we first define the threat model for our framework. We explain the model architecture for keyed image steganography and introduce different types of keys to ensure proper encryption of secret image. Finally, we discuss the strengths and limitations of our proposed **SecureImgStego**.

3.1 Threat Model

We consider a strong adversary with *white-box* access to our image steganography model, i.e., the adversary has complete access to the architecture, model

weights, and all hyper-parameters of the model. This adversary model subsumes any other weaker adversary that can train a surrogate of the original steganography model using an image dataset from similar distribution. However, the adversary does not have access to the keys used in our shuffling approach. Each `<sender, receiver>` pair shares a key that is assumed to be transmitted over a secure channel beforehand and therefore, the key is not accessible to the adversary. The carrier image is not shared over a secure channel. We assume that the adversary is a *man in the middle* which means that the adversary can eavesdrop the image sharing channel to capture the carrier image but it cannot drop or modify any image shared over the channel or inject any new image in the channel. The adversary may or may not have access to the actual cover image for each carrier image captured.

3.2 System Architecture

We design our deep neural network model for secure steganography system, `SecureImgStego`, which mainly consists of two components: an encoder on the sender end and a decoder on the receiver end. Fig. 2 summarizes the system architecture of our model. The encoder on the sender end consists of the following components: a shuffling layer, the preparation network (prep net), and the hiding network (hiding net). On the other hand, the decoder on the receiver end consists of the reveal network and the deshuffling layer.

Encoder: The *shuffling layer* takes the secret image (S) and the shared key (K , which represents a shuffling order) as inputs. We propose multiple shuffling techniques, i.e., pixel shuffling, multi-level block shuffling, and finally, a combination of key concatenation and block shuffling, and demonstrate their performance comparison in section 5. Pixel shuffling allows both spatial and channel shuffling where block shuffling allows only spatial shuffling. However, while pixel shuffling has a fixed level of depth, we can vary the depth of block shuffling by varying the block size and thus utilize the flexibility of multi-level block shuffling to minimize information loss in the reconstructed secret image (S'). Finally, a combination of key concatenation (similar to [20]) and multi-level block shuffling is applied to the secret image to further investigate the combined performance.

The *preparation network* is a convolutional neural network which generates high level features from the output of the shuffling layer. These high-level features are then embedded within the cover image (C) to generate the carrier image (C') using the *hiding network* which is also a convolutional neural network.

Decoder: The first component of the receiver end is the *reveal network*, implemented with a convolutional neural network as well, which takes the carrier image as input and extracts the encoded secret image. Finally, the *deshuffling layer* uses the shared secret key (specific to a `<sender, receiver>` pair) to retrieve the revealed secret image (S') from the output of the reveal network.

3.3 Training the Prep, Hiding, and Reveal Networks

In order to train the prep, hiding, and reveal convolutional neural networks, we use a subset of the Imagenet dataset as the training dataset (100000 images).

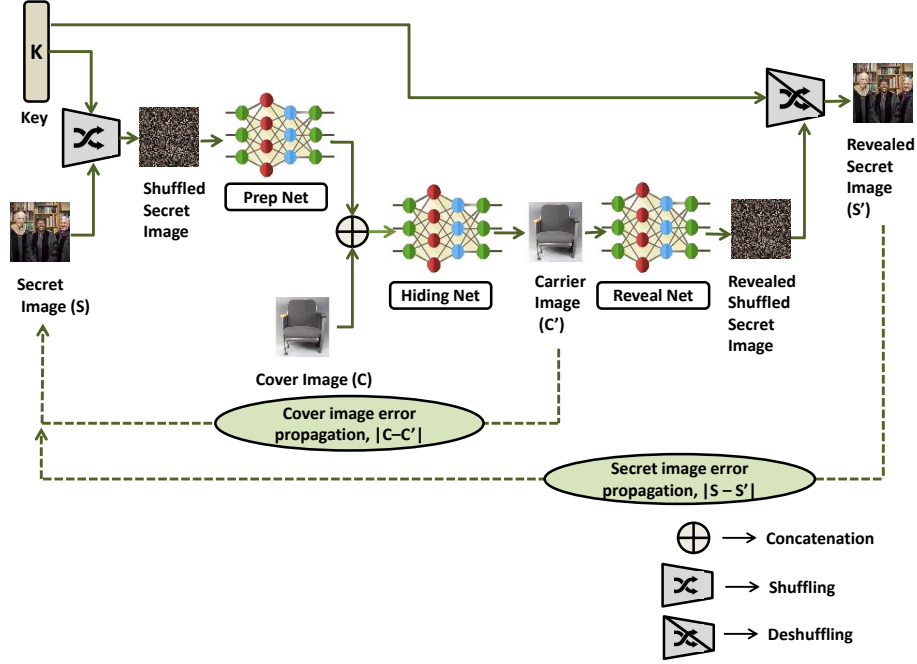


Fig. 2. System architecture of SecureImgStego. All five components, i.e., shuffling layer, prep net, hiding net, reveal net, and deshuffling layer, are trained simultaneously. Hence, during training, the error in the secret image is propagated into the whole network whereas the error in the cover image is propagated into the sender portion of the architecture (shuffling layer, prep, and hiding networks).

We randomly select half of these 100000 images as secret images and the rest 50000 images are used as the corresponding cover images. We generate a random shuffling key for each of these 50000 secret-cover image pairs. We train 4 separate models using this $\{K, S, C\}$ dataset for 4 different shuffling techniques, namely, pixel shuffling, one block (1B) shuffling, four block (4B) shuffling, and eight block (8B) shuffling. We also combine key concatenation [20] with 8B shuffling to understand if [20] provides any added advantage over our key shuffling encryption. For instance, in the case of 4B shuffling, each block consists of 4×4 pixels and represents 1 unit for shuffling. The sizes of keys for all models are described in Section 4.

The loss over the whole framework is calculated as follows:

$$L(C, C', S, S') = \|C - C'\| + \|S - S'\|$$

where, $\|C - C'\|$ represents the error in the carrier image when compared to the cover image and $\|S - S'\|$ represents the reconstruction error of the secret image. Note that, $\|C - C'\|$ error is not associated with the weights of the reveal

network which operates on the carrier image. Therefore, as shown in Fig. 2, the value of $\|C - C'\|$ is propagated into only the prep and hiding networks. Unlike $\|C - C'\|$, the $\|S - S'\|$ error is accumulated from all of the prep, hiding, and reveal networks, and is thus propagated into the whole framework. For all of our models, we use the same stopping criterion during training to ensure fairness in their comparison. We train all models for 50 epochs since according to our observation the training loss does not change significantly after 50 epochs.

Once the SecureImgStego model is trained, any `<sender, receiver>` pair that shares a secret key can use the model to securely transmit any secret image. Since we generated the shuffling keys randomly during training, our model is not overfitted to a particular shuffling order which is the case of Sharma et al. [28]. Moreover, a `<sender, receiver>` pair can select any of the 4 models trained with different shuffling techniques as appropriate for their secret image (only one shared key is sufficient across the models, more details in Section 5.4).

In section 4, we provide the details of the convolutional neural network architectures for the prep, hiding, and reveal networks. We also discuss the details of shuffling techniques in that section.

3.4 Discussion and Limitations

With the growing popularity of deep learning-based applications in our day-to-day activities, it is not surprising that deep learning-based steganography has now become the state-of-the-art [2,28,16,20,26,9]. However, deep learning-based solutions are often susceptible to different types of adversarial attacks [24]. To make matters worse, an adversary can also have white-box access to the model or can train a surrogate model with access to publicly available resources. Hence, it is important that we integrate enhanced security techniques with deep learning-based applications, e.g., deep learning-based image steganography which is used in critical applications, e.g., in military communications. To the best of our knowledge, this is the first research work that integrates key-based security in image steganography.

The main contribution of our model architecture is ensuring the requirement that any entity who wants to reveal the transmitted secret image needs to have access to the secret key. Any man in the middle attacker without access to the secret key cannot recover the secret image. According to our model architecture, even if an attacker has white-box access to the decoder network (or, a surrogate of the decoder network), the carrier image can not be successfully deshuffled without the secret key (see Section 5 for more details). Moreover, we leverage the flexibility of multi-level block shuffling encryption to optimize the performance by varying its depth for different images.

Since deep learning-based approaches for steganography can not ensure loss-less communication between sender and receiver, we could not utilize the state-of-the-art image encryption techniques. Instead, we deploy simple encryption techniques such as keyed pixel shuffling, multi-level block shuffling, and finally, a combination of key concatenation and block shuffling. Hopefully, our work takes the first step towards making deep learning-based image steganography secure and functional simultaneously.

4 Implementation Details of SecureImgStego

In this section, we discuss the implementation of SecureImgStego in details. More specifically, we discuss the dataset, different parameters and sizes of the keys used in our implementation, the platform used to carry out the experiments and finally, we provide a comprehensive summary of the architecture. To foster reproducibility, upon acceptance of the paper, we plan to release our codebase and the datasets used in the experiments.

4.1 Dataset

The cover and secret images used in our training and testing phases belong to the *Tiny-ImageNet* [1] dataset, a miniature version of the original ImageNet [5]. The Tiny-ImageNet consists of 110000 color images with shape of $\{64, 64, 3\}$. We randomly divide the image dataset into two subsets: a training of dataset of 100000 images and the rest 10000 images are used for testing. We follow the same process to divide the test dataset into secret and cover images as we mentioned in Section 3.3 for the training dataset. Hence, our train and test datasets do not introduce any bias in secret and cover image selection. Note that, we use the Tiny-ImageNet dataset instead of the original ImageNet dataset due to the constraints of CPU, GPU and memory usage in Google Colab platform.

4.2 Parameter Settings

We use Adam [17] as an optimizer algorithm to update the model weights during training. The learning rate is set to 0.001 and the batch size is set to 32.

In the case of key concatenation (similar to [20]), we randomly generate keys of shape $\{64, 64, 3\}$, i.e., the key size is 12288. For pixel shuffling, the key is the order of the pixels to be shuffled. Therefore, we generate keys by taking different permutations of the numbers in the set $\{1, 2, \dots, 12288\}$. Hence, the key size is also 12288. In the case of block shuffling methods, the key size can be expressed as p^2 , where $p = \frac{\text{image length}}{\text{block length}}$. For instance, in the case of 8B shuffling (where p is $\frac{64}{8} = 8$), the key size is $(8 \times 8) = 64$. For key concatenation + 8B shuffling method, the key size is $(8 \times 8) + (64 \times 64 \times 3)$ or 12352. The security guarantees for different key sizes are discussed in Section 5.6.

We define the *depth* (d) of block shuffling using the following formula: $d = \log_2 p$. For example, for 4B shuffling (where p is $\frac{64}{4} = 16$), the depth d is $\log_2 16 = 4$. Similarly, for 1B shuffling d is $\log_2 64 = 6$ and for 8B shuffling d is $\log_2 8 = 3$.

4.3 Environment Settings

SecureImgStego is implemented in a Jupyter iPython notebook on Google Colab platform. The host machine configuration is CPU: Intel(R) Xeon(R) CPU @ 2.20GHz (1 core, 2 thread), GPU: Tesla T4, 15 GB GDDR5 VRAM, CUDA version: 11.2, RAM: 12.72 GB, Disk Space: 107.77 GB, Python: 3.7.10, and OS: Ubuntu 18.04.5 LTS. We have used Tensorflow version 2.4.1 with Keras support.

4.4 Architecture

Our SecureImgStego architecture is inspired by Baluja [2] and Sharma et al. [28]. The preparation network is comprised of 2 convolutional layers with (50, 10, 5)

number of filters having kernel size $\{3 \times 3, 4 \times 4, 5 \times 5\}$ consecutively. Hiding network is composed of 6 convolutional layers with (50, 10, 5) number of filters having kernel size $\{3 \times 3, 4 \times 4, 5 \times 5\}$ consecutively. Again, the encoder consists of the preparation and hiding networks. Reveal network is constructed with 6 convolutional layers with (50, 10, 5) number of filters having kernel size $\{3 \times 3, 4 \times 4, 5 \times 5\}$ consecutively. These three networks are trained simultaneously. The shuffling and the deshuffling layers' operations are just opposite to each another. Both layers extract blocks from images and order them according to the provided key using tensorflow operation. Fig. 2 shows how these components interact among themselves to build SecureImgStego.

5 Experiment Results and Analysis

In this section, we present the result of our experiments with different keyed deep image steganography approaches. First, we show how different approaches' performance vary in terms of secret image reconstruction error, difference between the carrier and cover images, computation latency, and human perceptibility. We then demonstrate the robustness of our proposed block shuffling approach against surrogate model attack, cover image availability, and key brute-force attack. Due to space constraints, robustness of SecureImgStego against steganalysis is discussed in Appendix A.2

Table 1. Comparison among different image steganography approaches in terms of RMSE, SSIM, PSNR, and computation latency

Method	RMSE ↓		SSIM ↑		PSNR ↑		Latency ↓	
	Secret	Cover	Secret	Cover	Secret	Cover	Encoding	Decoding
Vanilla Deep Steganography [2]	8.34	8.59	0.94	0.90	30.23	29.72	5.91 s	4.08 s
Key Concat [20]	5.73	7.28	0.97	0.92	33.39	31.17	6.22 s	4.30 s
Pixel Shuffling	50.71	14.05	0.31	0.89	14.45	25.98	6.83 s	5.04 s
1B Shuffling	10.39	10.87	0.92	0.86	28.76	27.70	6.4 s	4.48 s
4B Shuffling	5.72	8.89	0.96	0.89	33.51	29.37	6.29 s	4.75 s
8B Shuffling	7.07	8.46	0.95	0.90	31.59	29.85	6.16 s	4.30 s
Key Concat [20] + 8B Shuffling	8.33	11.34	0.95	0.90	30.00	27.90	6.40 s	4.48 s





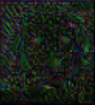
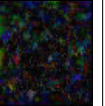




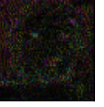
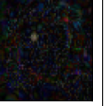



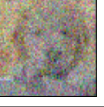
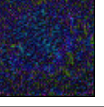
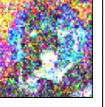




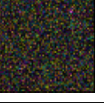
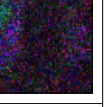


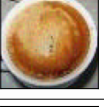
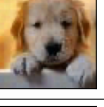




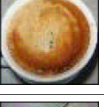
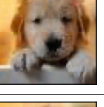

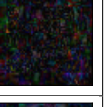


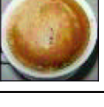
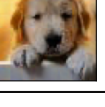
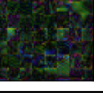

5.1 Performance Analysis

We depict the quality measures such as RMSE, SSIM, PSNR, and computation latency for different steganography methods in Table 1. For space constraints, the mathematical definitions of RMSE, SSIM, and PSNR are provided in Appendix A.1. Each of the quality measure of different steganography methods presented in Table 1 is calculated by taking the average of 10 runs on the test dataset. It shows 4B and 8B shuffling perform better than vanilla deep steganography method in terms of image quality. This is because CNN convolves around images to learn features from blocks or clusters of pixels and thus, additional block creation has become an advantage to CNN here. For the same reason, 1B and pixel shuffling perform worse as these shufflings scatter pixels indiscriminately, and therefore, CNN fails to extract useful features. We have also compared our methods with key concatenation, an approach for text data [20], to show how

this works for images. In our key concatenation implementation, we concatenate (64,64,3) shaped byte key with input image and it performs better in image quality.

However, in Table 3, we see that key concat performs unsatisfactorily in decryption. Key Concat + 8B shuffling has no advantage over 8B shuffling, rather it exacerbates image quality. We report encoding-decoding time for the entire test dataset. Hence, the required time is proportional to the depth of shuffling. As the depth of shuffling increases, the number of possible permutations increases and thus takes more computational time. Pixel shuffling has the highest latency since it shuffles along both spatial and channel dimensions.

Table 2. Visual analysis of different steganography methods’ performances





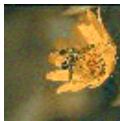

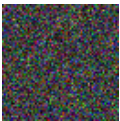
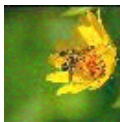





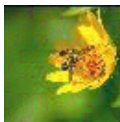
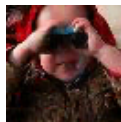
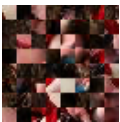
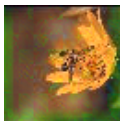

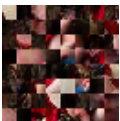
Model	Cover	Secret	Encoded Cover	Decoded Secret	Diff Cover $\times 5$	Diff Secret $\times 5$
Vanilla Deep Steganography [2]						
Key Concat [20]						
Pixel Shuffling						
1B Shuffling						
4B Shuffling						
8B Shuffling						
Key Concat [20] + 8B Shuffling						

5.2 Visual Analysis

Table 2 shows the visual representation of secret and cover images before and after encoding along with their amplified differences. In the cases of [2] and [20], differences between the original cover and the encoded cover (carrier) images

with $5\times$ enhancement reveal the shape of the secret image (i.e., the face of a dog in this example). Note that, the results with key concat [20] is no better than vanilla deep steganography in terms of concealing the secret image from the difference between cover and carrier images. In contrast, our keyed shuffling approaches resolve this issue of leaking secret image from carrier image. Among all the keyed approaches, pixel shuffling performs the worst in terms of reconstructing the secret image mainly due to the fact that pixel shuffling allows both spatial and channel shuffling. More details on the limitations of pixel shuffling is provided in Section 5.1.

Table 3. Effect of adversary generated random Key




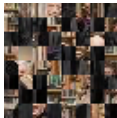






Secret Image	Model	Carrier Image	Revealed Secret Image	
			With Correct Key	With Random Key
	Key Concat [20]			
	Pixel Shuffling			
	1B Shuffling			
	4B Shuffling			
	8B Shuffling			
	Key Concat [20]+ 8B Shuffling			

5.3 Sensitivity to Random Key

As illustrated in Fig 1, an adversary with access to the decoder (or, a surrogate of decoder) cannot retrieve the original secret image if we design a proper keyed image steganography. In Table 3, we show how the adversary generated random keys perform in revealing the secret image when compared to the correct keys.

Note that, Key concat [20] performs poorly in the case of image data although the same method works for text data. This is because in the text data when the ASCII value of a character changes from 65 to 64, the character changes to ‘@’ from ‘A’ whereas in the case of RGB colored image if the R component changes from 65 to 64, the change is imperceptible to the human eyes. As a result, the adversary can reveal a blurred version of the secret image with a random key which compromises security. Although pixel shuffling performs poorly even with correct key, for all other shuffling techniques, we can see significant differences between the secret images revealed with correct and random keys which demonstrates the security of our proposed models against an advanced adversary who has white-box access to the decoder but does not know the correct key.

Table 4. Effect of different depth values in multi-level block shuffling

Secret Image	Model	Depth	Carrier Image	Revealed Secret Image	
				With Correct Key	With Random Key
	8B Shuffling	3			
	4B Shuffling	4			
	1B Shuffling	6			




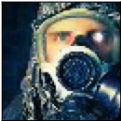
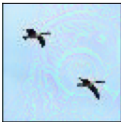
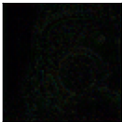


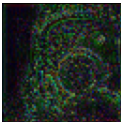

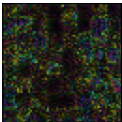

5.4 Adaptive Depth for Multi-level Block Shuffling

Our empirical analysis shows that the appropriate level or depth of block shuffling is highly correlated with the contents of the secret image. If a secret image contains significant information after d -depth block shuffling, the sender should instead use block shuffling with a depth higher than d . As demonstrated in Table 4, the revealed secret image derived with a random key in the case of 8B shuffling leaks crucial information (block at row 1, column 4 almost reveals the face of the left-most person in the original secret image). The revealed secret image with a random key in the case of 4B shuffling still reveals human eyes whereas 1B shuffling (highest depth, $d = 6$) makes it impossible for the adversary to comprehend the content of the secret image. However, a simple analysis of Table 1 and 3 reveals that the granularity of block shuffling results in a trade-off between reconstructed image quality and security. Hence, the sender can select the appropriate depth for block shuffling according to the quality and security

requirements of a secret image. *Note that, in our experiments, we collect secret images from the ImageNet dataset. However, in practice, the sender is free to craft secret images intelligently so that they can achieve the best of both image quality and security.*

It is important to note that SecureImgStego requires only one shared key between each `<sender, receiver>` pair while allowing them to switch to multiple NB shuffling models for adaptivity, where, for instance, $N = 4$ represents 4B shuffling. More specifically, it is sufficient for a `<sender, receiver>` pair to have a shared key for the model with the smallest block shuffling. For example, if the `<sender, receiver>` pair has a shared key of length 4096 for 1B shuffling, this key can also be used in 4B shuffling where the intended key size is 256. The pair can extract the integers $1, \dots, 256$ from the original shared key without altering their order. Similarly, the pair can derive the key for 8B shuffling of size 64 in a similar way.

Table 5. Effect of Encryption

Vanilla Deep Steganography [2]		8B Shuffling	
Original Cover	Revealed Secret	Original Cover	Revealed Secret
			
Encoded Cover	Diff Cover	Encoded Cover	Diff Cover
			
Diff Cover x5	Diff Cover Grey x5	Diff Cover x5	Diff Cover Grey x5
			

5.5 Availability of Cover Image

The deep image steganography method proposed in [2] requires that the adversary does not have access to the original cover image. Otherwise, it poses a significant threat that the secret image could be partially revealed by the adversary by taking the difference between the original cover image and carrier image. Results in Table 5 confirm this limitation of [2] since the Diff cover (i.e., $\|C - C'\|$) with enhancement partially reveals the secret image (a masked human in this example). Our keyed shuffling methods do not have such limitations since no visual information can be retrieved even in the gray-scale diff enhancement. Therefore, our keyed shuffling methods are more secure than the previous methods when the original cover image is available to the adversary.

5.6 Key Space and Security Guarantees

In the case of 8B shuffling, the number of blocks to be shuffled is $p^2 = \frac{\text{image length}}{\text{block length}} \times \frac{\text{image width}}{\text{block width}} = \frac{64}{8} \times \frac{64}{8} = 64$. Thus the key size in this case is also 64. Since the adversary does not have access to the shuffling key, it may try to launch a brute-force attack to learn the shared secret key of a targeted **<sender, receiver>** pair. This requires the adversary to test with $64!$ combinations ($\sim 1.268869 \times 10^{89}$) in the worst case to learn the correct key for 8B shuffling (given that 8B shuffling is the optimal choice for the sender to transmit the secret image). Even if the adversary has access to a computing resource which can test 10^{17} keys in 1 second, it would still need 4.023558×10^{64} years for the adversary to learn the secret key. In a coarser-grained shuffling, i.e., when depth of shuffling is lower and block length is higher, the key size would be smaller and therefore, the adversary would need less time to recover the key (e.g., in 16B shuffling). In contrast, if we consider a more fine-grained shuffling, e.g., in 1B shuffling, depth of shuffling is higher and block length is lower. The key size is $p^2 = \frac{64}{1} \times \frac{64}{1} = 4096$ which significantly increases the number of key combinations and the time required for the adversary to learn the key. In summary, the security guarantee for the key in our model is proportional to the depth of block shuffling or in other words, inversely proportional to the block length.

6 Conclusion and Future Work

In this paper, we have introduced a novel keyed image steganography model based on deep convolutional neural networks. Our encryption layer employs the secret key which adds an additional layer of security to the deep convolutional neural network. Therefore, our image steganography model ensures secure communication between sender and receiver **despite the assumption that the adversary has access to a surrogate model**. We have demonstrated that our *provably secure* steganography system is capable of concealing full-size color image into another image without losing visual characteristics and can ensure *confidentiality* even when the carrier image is available to the attacker. Our findings show the tradeoff between the performance and level of encryption in different granularity of block shuffling. Our multi-level block shuffling method facilitates adjust-ability to different depth of block shuffling according to specific image.





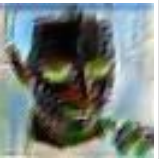









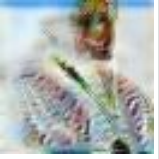
One future direction of our work is to extend it to asymmetric key based-SecureImgStego to cover the cases where the **<sender, receiver>** pair does not already have a shared secret key. Additionally, our approach for secure image steganography can also be extended for audio and video data. Finally, hiding multiple secret images in a single cover image using SecureImgStego will also be an interesting extension.

7 Attack Demos

Our experiment shows that the deep steganographic model entirely on its own cannot ensure enough security, it needs to be backed up with encryption. In an

attempt to demonstrate this, we devise three adversaries considering the variation of access and knowledge an attacker may have in real-world attacks. We divide our whole dataset randomly into two equal portions so that the vanilla model and adversary models have no overlap in the training dataset. Our chosen three adversaries, i.e., white-box, grey-box, and black-box, are designed on different complexity levels. White-box adversary is the most knowledgeable one with all information about the architecture and hyperparameters of the vanilla model. It is to be noted that though both the vanilla model and white-box adversary are trained on the same architecture, their diversity in the training dataset makes them different. Grey-box adversary [A.3](#) is aware of the 3 components in the vanilla model, i.e, prep net, reveal net, and hiding net. However, he is ignorant of the filter numbers, kernel size, and layer structures used in the vanilla model. Hence, he replicates a popular image classification model architecture, InceptionV3 [\[30\]](#), and repeats it as prep, reveal, and hiding net. The black-box adversary [A.3](#) is the most uninformed one and even blind about the 3 parts of the vanilla model. Therefore, he imitates another well-known image classification model, VGG16 [\[27\]](#), twice to construct just one encoder and one decoder. Table [6](#) presents the results of such successful attacks where decoded secret images by three adversaries are conspicuous enough to reveal the secret information.









Table 6. Attacks in vanilla deep steganography

Secret Image	Cover Image	Decoded Secret Image		
		White-box	Grey-box	Black-box
				
				
				

8 Steganalysis with Deep Steganography

As defined in 2.1, steganalysis is the process to label an image either as a carrier (image containing hidden information) or as an unperturbed (image with no hidden information). In terms of this steganalysis, Baluja [2] demonstrates the robustness of the deep steganographic method as it succeeds to dodge a traditional lower bit seeking steganographic tool named StegExpose [4]. However, we examine that an adversary can achieve the same objective of differentiating between the carrier and unperturbed image by employing the deep steganography model itself. For instance, while being unsure whether an image contains secret information or not, the attacker can pass the image to a deep steganographic model. Table 7 manifests if the input image is an unperturbed one, its decoded secret image is somewhat meaningless which ascertains that the input image has no information hidden in it. On the other hand, in the case of a carrier image, the adversary will get a meaningful image and thus can conclude that the image has secret information embedded. It prompts our thought that an attacker may detect the presence of secret information in an image quite readily, therefore, the integration of encryption is necessary to establish the defense-in-depth.

Table 7. Steganalysis with Deep Steganography

Input Type	Input Image	Decoded Secret Image		
		White-box	Grey-box	Black-box
Unperturbed				
Carrier				

References

1. Tiny imagenet - <http://cs231n.stanford.edu/tiny-imagenet-200.zip>
2. Baluja, S.: Hiding images in plain sight: Deep steganography. In: Advances in Neural Information Processing Systems. pp. 2069–2079. Long Beach, CA, USA (Dec 4-Dec 9 2017)
3. Bharti, S., Behal, S., Sharma, V.: Security enhancements for high quality image transaction with hybrid image steganography algorithm. In: 2018 Second International Conference on Computing Methodologies and Communication (ICCMC). pp. 162–169. IEEE (2018)
4. Boehm, B.: Stegexpose-a tool for detecting lsb steganography. arXiv preprint arXiv:1410.6656 (2014)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
6. Ding, Y., Wu, G., Chen, D., Zhang, N., Gong, L., Cao, M., Qin, Z.: Deepedn: A deep-learning-based image encryption and decryption network for internet of medical things. IEEE Internet of Things Journal **8**(3), 1504–1518 (2021). <https://doi.org/10.1109/JIOT.2020.3012452>
7. Duan, X., Guo, D., Liu, N., Li, B., Gou, M., Qin, C.: A new high capacity image steganography method combined with image elliptic curve cryptography and deep neural network. IEEE Access **8**, 25777–25788 (2020). <https://doi.org/10.1109/ACCESS.2020.2971528>
8. Duan, X., Guo, D., Liu, N., Li, B., Gou, M., Qin, C.: A new high capacity image steganography method combined with image elliptic curve cryptography and deep neural network. IEEE Access **8**, 25777–25788 (2020)
9. Duan, X., Jia, K., Li, B., Guo, D., Zhang, E., Qin, C.: Reversible image steganography scheme based on a u-net structure. IEEE Access **7**(6), 9314–9323 (2019)
10. El-Khamy, S.E., Korany, N.O., Mohamed, A.G.: A new fuzzy-dna image encryption and steganography technique. IEEE Access **8**, 148935–148951 (2020)
11. Fridrich, J., Goljan, M.: Practical steganalysis of digital images: state of the art. In: Security and Watermarking of Multimedia Contents IV. vol. 4675, pp. 1–13. International Society for Optics and Photonics (2002)
12. Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: International Conference on Machine Learning. pp. 201–210. PMLR (2016)
13. Hu, F., Wang, J., Xu, X., Pu, C., Peng, T.: Batch image encryption using generated deep features based on stacked autoencoder network. Mathematical Problems in Engineering **2017** (2017)
14. Kessler, G.C.: An overview of steganography for the computer forensics examiner. Forensic science communications **6**(3), 1–27 (2004)
15. Kessler, G.C., Hosmer, C.: An overview of steganography. In: Advances in Computers, vol. 83, pp. 51–107. Elsevier (2011)
16. Khan, N., Haan, R., Boktor, G., McComas, M., Daneshi, R.: Steganography gan: Cracking steganography with cycle generative adversarial networks. arXiv preprint **arXiv:2006**(8) (2020)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

18. Kreuk, F., Adi, Y., Raj, B., Singh, R., Keshet, J.: Hide and speak: Deep neural networks for speech steganography. arXiv preprint **arXiv:1902**(83) (2019)
19. Li, B., He, J., Huang, J., Shi, Y.Q.: A survey on image steganography and steganalysis. *Journal of Information Hiding and Multimedia Signal Processing* **2**(2), 142–172 (2011)
20. Li, Z., Han, G., Guo, S., Hu, C.: Deepkeystego: Protecting communication by key-dependent steganography with deep networks. In: 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). pp. 1937–1944 (2019). <https://doi.org/10.1109/HPCC/SmartCity/DSS.2019.00267>
21. Li, Z., Han, G., Guo, S., Hu, C.: Deepkeystego: Protecting communication by key-dependent steganography with deep networks. In: 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). pp. 1937–1944. IEEE (2019)
22. Neeta, D., Snehal, K., Jacobs, D.: Implementation of lsb steganography and its evaluation for various bits. In: 2006 1st International Conference on Digital Information Management. pp. 173–178. IEEE (2006)
23. Ozer, H., Avcibas, I., Sankur, B., Memon, N.D.: Steganalysis of audio based on audio quality metrics. In: Security and Watermarking of Multimedia Contents V. vol. 5020, pp. 55–66. International Society for Optics and Photonics (2003)
24. Papernot, N., McDaniel, P., Sinha, A., Wellman, M.P.: Sok: Security and privacy in machine learning. In: 2018 IEEE European Symposium on Security and Privacy (EuroS P). pp. 399–414 (2018). <https://doi.org/10.1109/EuroSP.2018.00035>
25. Sarmah, D.K., Kulkarni, A.J.: Improved cohort intelligence—a high capacity, swift and secure approach on jpeg image steganography. *Journal of information security and applications* **45**(3), 90–106 (2019)
26. Sharma, K., Aggarwal, A., Singhania, T., Gupta, D., Khanna, A.: Hiding data in images using cryptography and deep neural network. arXiv preprint arXiv:1912.10413 (2019)
27. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014). <https://doi.org/10.48550/ARXIV.1409.1556>, <https://arxiv.org/abs/1409.1556>
28. Singhania, T., Aggarwal, A., sharma, k., Khanna, A., Gupta, D.: Hiding data in images using cryptography and deep neural network **1**, 143–162 (12 2019). <https://doi.org/10.33969/AIS.2019.11009>
29. Subhedar, M.S., Mankar, V.H.: Current status and key issues in image steganography: A survey. *Computer science review* **13**, 95–113 (2014)
30. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision (2015). <https://doi.org/10.48550/ARXIV.1512.00567>, <https://arxiv.org/abs/1512.00567>
31. Tancik, M., Mildenhall, B., Ng, R.: Stegastamp: Invisible hyperlinks in physical photographs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2117–2126. CVPRVirtual (Jun 16–jun 18 2020)
32. Wengrowski, E., Dana, K.: Light field messaging with deep photographic steganography. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1515–1524. Long Beach, CA, USA (Jun 16–jun 20 2019)
33. Wu, P., Yang, Y., Li, X.: Stegnet: Mega image steganography capacity with deep convolutional network. *Future Internet* **10**(6), 54 (2018)

A Appendix

A.1 Reconstruction Metrics

The equation of Mean Squared Error (MSE) between secret image (S) and revealed secret image (S') is defined as follows:

$$\text{MSE}(S, S') = \frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n (S_{ij} - S'_{ij})^2 \quad (1)$$

$$\text{RMSE}(S, S') = \sqrt{\text{MSE}(S, S')} \quad (2)$$

where: m = number of rows in secret image of S or S'
 n = number of columns in image of S or S'
 S_{ij} = pixel value from S
 S'_{ij} = pixel value from S'

Structural Similarity Index Measure (SSIM) is a mapping of visibility error to structural information. Its equation is as follows :

$$\text{SSIM}(S, S') = \frac{(2\mu_S\mu_{S'} + C_1) + (2\sigma_{SS'} + C_2)}{(\mu_S^2 + \mu_{S'}^2 + C_1)(\sigma_S^2 + \sigma_{S'}^2 + C_2)} \quad (3)$$

where: μ_S = average of S
 $\mu_{S'}$ = average of S'
 σ_S^2 = variance of S
 $\sigma_{S'}^2$ = variance of S'
 $\sigma_{SS'}$ = co-variance of S and S'
 $c_1 = (k_1 L)^2$
 $c_2 = (k_2 L)^2$
 L = dynamic range of the pixel-values
 $k_1 = 0.01$
 $k_2 = 0.03$

Peak signal-to-noise ratio (PSNR) is a logarithmic ratio between maximum possible pixel value of image and MSE. The equation of PSNR is given below :

$$\text{PSNR}(S, S') = 10 \log_{10} \left(\frac{(\max(S))^2}{\text{MSE}(S, S')} \right) \quad (4)$$

where: m = number of rows in image of S or S'
 n = number of columns in image of S or S'
 S_{ij} = pixel value from S
 S'_{ij} = pixel value from S'

The corresponding equations of RMSE, SSIM, and PSNR for cover image (C) and carrier image (C') are equivalent.

A.2 Steganalysis

To investigate the performance of existing approaches and the proposed secure deep steganography approaches against steganalysis, we use *StegExpose* [4], a steganalysis tool popular among researchers and practitioners which has also been used in [2] and [20]. We use the default setting of the StegExpose tool in our experiments to plot the ROC curves as shown in Fig. 3. We find that introducing keyed deep image steganography does not have any significant impact on the performance of StegExpose when compared to the vanilla deep steganography [2] or key concatenation [20]. Therefore, our proposed **SecureImgStego** is as robust as other image steganography approaches against steganalysis.

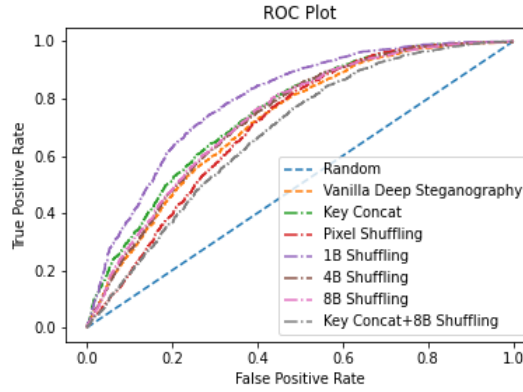


Fig. 3. Performance of existing and proposed approaches against StegExpose

A.3 Adversary Model Diagram

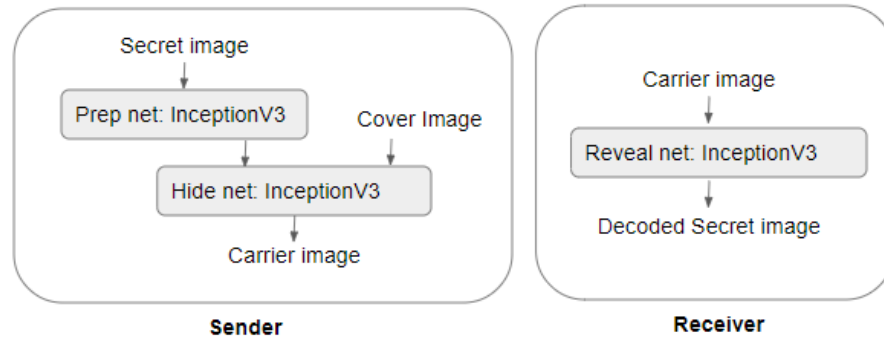


Fig. 4. Block diagram of grey-box adversary



Fig. 5. Block diagram of black-box adversary