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

Trishna Chakraborty
University of California, Irvine
trishnac@uci.edu

Imranur Rahman
North Carolina State University
irahman3@ncsu.edu

Hasan Murad
Chittagong University of Engineering and Technology
muradbuetcse13@gmail.com

Md Shohrab Hossain
Bangladesh University of Engineering and Technology
mshohrabhossain@cse.buet.ac.bd

Shagufta Mehnaz
The Pennsylvania State University
smehnaz@psu.edu

Abstract—Steganography ensures secure transmission of digital messages, including image steganography where a secret image is hidden within a non-secret cover image. Deep learning-based methods in image steganography have recently gained popularity but are vulnerable to various attacks. An adversary with varying levels of access to the vanilla deep steganography model can train a surrogate model using another dataset and retrieve hidden images. Moreover, even when uncertain about the presence of hidden information, the adversary with access to the surrogate model can distinguish the carrier image from the unperturbed one. Our paper includes such attack demonstrations that confirm the inherent vulnerabilities present in deep learning-based steganography. Deep learning-based steganography lacks lossless transmission assurance, rendering sophisticated image encryption techniques unsuitable. Furthermore, key concatenation-based techniques for text data steganography fall short in the case of image data. In this paper, we introduce a simple yet effective keyed shuffling approach for encrypting secret images. We employ keyed pixel shuffling, multi-level block shuffling, and a combination of key concatenation and block shuffling, embedded within the model architecture. Our findings demonstrate that the block shuffling-based deep image steganography has negligible error overhead compared to conventional methods while providing effective security against adversaries with different levels of access to the model. We extensively evaluate our approach and compare it with existing methods in terms of human perceptibility, key sensitivity, adaptivity, cover image availability, keyspace, and robustness against steganalysis.

I. INTRODUCTION

In this modern era, ensuring privacy and security in computer and internet usage is a significant challenge. Steganography, the technique of concealing a secret message within a non-secret image, has emerged as a successful method for secure transmission of confidential digital content over unreliable media. It has a wide range of applications, including military communications, protecting authorship information, and preventing data alteration. For a basic introduction to steganography systems and their applications, refer to [1], [2].

Prior research. The traditional approaches for steganography [3], [4] 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 [5]–[7]. Baluja [5] 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 [8].

Limitations of existing approaches and challenges. Despite advancements in embedding secret messages within carrier images, traditional steganography techniques can be identified through statistical analysis [9], [10]. Deep image steganography methods are also vulnerable to adversaries retrieving secret images from carriers, even without access to the original training dataset. This makes existing deep image steganography models susceptible to attacks, including man-in-the-middle (MITM) attacks. Combining cryptographic techniques with steganography as a pre-processing step has been attempted to address these security concerns [11]–[15]. However, previous approaches like fixed shuffling pre-processing [12] lack support for secure steganography with multiple sender-receiver pairs and are vulnerable to white-box attacks. Additionally, applying advanced image encryption techniques directly to deep learning-based steganography models is challenging due to potential loss in carrier image transmission. Li et al. [11] introduced a key concatenation-based encryption technique for secure transmission of text data in deep steganography networks, but it is inadequate for image data. Therefore, the limited applicability of existing research in the image domain presents a significant challenge for *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-
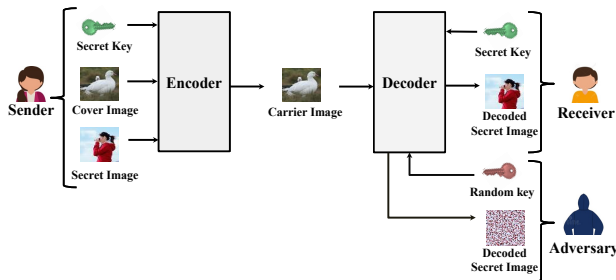
Fig. 1. Sender and receiver share a secret key to encode and decode the secret image, respectively. An adversary with access to the decoder (or, a surrogate model of the decoder) may attempt to decode the carrier image with a random key.

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 [5], i.e., a stenganography model without any keyed encryption. Moreover, we present extensive evaluation and compare different types of key usages along with their trade-offs between security and utility.

Approach. Our image steganography framework, shown in Fig. 1, comprises an encoder and a decoder implemented using CNN. It utilizes a shared secret key between the sender and receiver. The encoder encrypts the secret image using the shared key and embeds it into the cover image for transmission. The decoder uses the shared key to decrypt the carrier image and recover the secret image. We employ various shared keys, including keyed pixel shuffling, multi-level block shuffling, and a combination of key concatenation and block shuffling. We train, validate, and test our model using the Tiny ImageNet [16] dataset for secret and cover images. Our results demonstrate that the inclusion of keyed block shuffling in the deep learning-based steganography model introduces minimal error overhead while ensuring secure transmission. It outperforms vulnerable vanilla deep steganography [5] approaches susceptible to surrogate model attacks. Our framework is robust against statistical analysis and steganalysis methods, such as StegExpose [17]. The flexibility of multi-level block shuffling encryption, adapting its depth for different secret images, achieves optimal performance in SecureImgStego.

Contributions. The key research contributions are:

- We expose the inherent vulnerabilities in deep image steganography, demonstrating real attacks where significant information about the secret image is disclosed to an attacker, regardless of their access level to the deep steganography model (full, partial, or zero).
- 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. It ensures secure communication despite the assumption that the adversary owns a surrogate model of the original one.
- 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 extensively evaluate SecureImgStego and compare it with existing methods in terms of human perceptibility, key sensitivity, adaptivity, cover image availability, keyspace, and steganalysis robustness.

We organize the rest of the paper as follows. In Section II, we discuss the related works and identify their limitations. In Section III, we explain SecureImgStego along with the threat model. Section IV presents the implementation details. In Section V, we extensively evaluate SecureImgStego and report our findings. Finally, we discuss some future research directions and conclude in Section VI.

II. BACKGROUND AND RELATED WORK

In this section, we provide an overview of the key terms used in steganography and cryptography. We review related works, highlight their limitations, and conduct a gap analysis based on the current state-of-the-art research.

A. Background

Steganography conceals confidential data within non-confidential data to avoid detection by intermediaries. It can involve various types of data such as text, images, audio, or video. The process typically involves three components: the secret object (payload), the cover object, and the carrier object. The secret object is the information to be hidden, embedded within the cover object. The carrier object, a combination of the secret and cover objects, is publicly available and should visually resemble the cover object to maintain secrecy. Steganalysis refers to the detection or recovery of the secret object from the carrier object. In cryptographic systems, encryption keys are used to secure information. Symmetric cryptography employs a shared private key for both encryption and decryption, while asymmetric cryptography uses a public key for encryption and a private key for decryption. In our work, we utilize symmetric cryptography for the secure transmission of the secret object.

B. Related Work

To systematize the diverse steganography research, we categorize the existing 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 [3], [4]. The least significant bits (LSBs) replacement is the most popular among the practitioners in image steganography [18]. 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 [9], [10].

Deep Steganography. Automatic feature detection capability of deep neural networks has given rise to deep steganography. Baluja [5] proposes an end-to-end deep image steganography framework that 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. [19] 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. [20] encode 56-bit hyperlink bitstrings into images. Kreuk et al. [21] 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. [22] use the Cohort Intelligence (CI) algorithm to reduce computational cost.

Encrypted Steganography. There exists a number of research works [23]–[25] that focus on image cryptography. Ding et al. [23] propose a cycle GAN-based approach for medical images where parameters of the network are considered as key whereas Gilad-Bachrach et al. [24] 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. [11] 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, fails to decode the hidden text message, and ends up with random strings. Duan et al. [26] 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. [12] 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.

C. Gap Analysis

Existing image steganography works focus on improving model architecture but neglect security issues. Adversaries can train surrogate models and retrieve secret images without access to the original dataset or knowledge of the model architecture. Unfortunately, the solution proposed in [11] only applies to text data and lacks information on extending it to handle image data. In Section V-E, we demonstrate why the key concatenation approach fails when dealing with images. Duan et al. [26] 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 [5] 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.
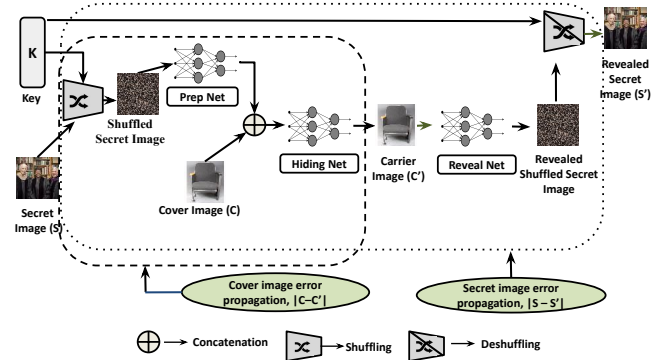

Fig. 2. System architecture of SecureImgStego. All five components, i.e., shuffling layer, prep net, hiding net, reveal net, and deshuffling layer, are trained simultaneously. Hence, during training, the error in the secret image is propagated into the whole network whereas the error in the cover image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

This limitation has been addressed by Sharma et al. [12] where they hide a scrambled version of the secret image into the cover image. However, their model's constant pixel scrambling order allows adversaries with a surrogate model to easily decode the secret image. Furthermore, their approach is vulnerable to known-plaintext attacks, where obtaining the scrambling order of one secret image enables the retrieval of all encoded secret images by the adversary.

Our novel approach introduces a simple, fast, and effective symmetric keyed shuffling method. It prevents the retrieval of secret images by adversaries, even if they train surrogate models or have access to the actual cover image. Furthermore, we prioritize preserving the utility of the steganography model, resulting in negligible increase in error overhead compared to the insecure vanilla deep steganography model [5].

III. PROPOSED SecureImgStego MODEL

In this section, we define the threat model for our framework, explain the model architecture for keyed image steganography, introduce various types of keys for secure encryption of the secret image, and discuss the strengths and limitations of our proposed SecureImgStego.

A. Threat Model

Considering the varying knowledge and access of real-world attackers, we classify them into three categories: white-box, grey-box, and black-box adversaries. The white-box adversary possesses complete knowledge of the architecture and hyperparameters of the vanilla model. The grey-box adversary has partial knowledge of the model architecture, while the black-box adversary has no knowledge of it. None of these adversaries have access to the original dataset (refer to Section V-A) Additionally, the adversary owns no 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 transmitted over

a secure channel, and we assume the adversary is a passive eavesdropper (man in the middle) who can observe the channel but cannot modify or inject any images. The adversary may or may not have access to the actual cover image for each carrier image captured. Some may contend that our threat model diverges from the traditional concept of steganography, where the adversary remains unaware of whether an image is a carrier or unaltered. However, we maintain that assuming an entirely ignorant adversary is impractical in real-world scenarios. To support this standpoint, we present compelling evidence (refer to Section V-B) that even when the adversary is uncertain about the presence of secret information, they can employ a deep learning-based surrogate decoder to swiftly cross-check for the existence of any concealed data within a short timeframe.

B. 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 V. 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 [11]) 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 output of the shuffling layer. These high-level features are then embedded within the cover image (C) to generate 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 from the output of the reveal network.

C. Training the Prep, Hiding, and Reveal Networks

To train the convolutional neural networks for the prep, hiding, and reveal stages, we utilize a subset of the Imagenet dataset comprising 100000 images. Among these images, we randomly select 50000 as secret images and the remaining 50000 as cover images. For each secret-cover image pair, we generate a random shuffling key. Four separate models are trained using this $\{K, S, C\}$ dataset, employing different shuffling techniques: pixel shuffling, one block (1B) shuffling, four block (4B) shuffling, and eight block (8B) shuffling. Additionally, we explore the combination of key concatenation [11] with 8B shuffling. The sizes of the keys for all models are detailed in Section IV.

The loss over the whole framework is: $L(C, C', S, S') = ||C - C'|| + ||S - S'||$. In the proposed framework, the error in the carrier image compared to the cover image, represented as $||C - C'||$, is independent of the reveal network's weights. This error is only considered in the prep and hiding networks, as shown in Fig. 2. On the other hand, the reconstruction error of the secret image, $||S - S'||$, accumulates from all three networks (prep, hiding, and reveal) and affects the entire framework. To ensure fairness in comparison, all models follow the same stopping criterion during training. We train each model for 50 epochs, as our observations indicate that the training loss stabilizes after this point.

Once the SecureImgStego model is trained, any <sender, receiver> pair with a shared secret key can utilize the model for secure transmission of any secret image. Our model's versatility is ensured as the shuffling keys were randomly generated during training, avoiding overfitting to a specific shuffling order, unlike Sharma et al. [12]. The pair can select the appropriate model among the four trained models with different shuffling techniques, requiring only one shared key across the models (further details in Section V-F). Section IV provides comprehensive information about the convolutional neural network architectures for the prep, hiding, and reveal networks, as well as details about the employed shuffling techniques.

D. 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 [5], [7], [8], [11], [12], [12]. However, deep learning-based solutions are often susceptible to different types of adversarial attacks [27]. To make matters worse, an adversary can train a surrogate model with full, partial, or no access to the original deep steganography model.

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 symmetric 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 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 V for more details). Moreover, we leverage the flexibility of multi-level block shuffling encryption to optimize performance by varying its depth for different images.

Deep learning-based steganography methods lack lossless communication guarantees, preventing the use of advanced image encryption techniques. Instead, we employ simple encryption techniques such as keyed pixel shuffling, multi-level block shuffling, and 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.

IV. 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, and finally, we provide a comprehensive summary of the architecture.

A. Dataset

The training and testing phases of our work utilize the *Tiny-ImageNet* [16] dataset. It consists of $110,000$ color images with dimensions of $\{64, 64, 3\}$. We randomly split the dataset into a training set of $100,000$ images and a test set of $10,000$ images. Similar to our process for the training dataset mentioned in Section III-C, we divide the test dataset into secret and cover images. This ensures unbiased selection of secret and cover images in both training and testing datasets.

B. Parameter Settings

We use Adam [28] 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 [11]), 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 $\{1, 2, \ldots, 12288\}$. Hence, the key size is also 12288. In the case of block shuffling methods, the key size can be expressed as p^2, where $p = \frac{image\ length}{block\ length}$. 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 V-J. We define the *depth* (d) of block shuffling using the following formula: $d = \log_2 p$. For example, for 4B shuffling (where p is $64/4 = 16$), the depth d is $\log_2 16 = 4$. Similarly, for 1B shuffling d is $\log_2 64 = 6$ and for 8B shuffling d is $\log_2 8 = 3$.

C. Architecture

Our SecureImgStego architecture is inspired by Baluja [5] and Sharma et al. [12]. 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 operations in the shuffling and deshuffling layers are just opposite to each other. 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.

D. Block Shuffling Procedure

Formally speaking, we consider an image of dimension $\{H \times W \times C\}$ is divided into some blocks (each block is denoted by b): $\begin{bmatrix} b_{11} & \ldots & b_{1k} \\ \vdots & \ddots & \\ b_{k1} & \ldots & b_{kk} \end{bmatrix}$. We then serialize the blocks in a row-wise fashion: $\begin{bmatrix} b_{11} \ldots b_{1k} \ldots b_{k1} \ldots b_{kk} \end{bmatrix}$. We use the key of the same length to convert this serialization to a different mapping: $\begin{bmatrix} b'_{11} \ldots b'_{1k} \ldots b'_{k1} \ldots b'_{kk} \end{bmatrix}$. After that, we de-serialize this in a row-wise fashion and construct the desired shuffled image: $\begin{bmatrix} b'_{11} & \ldots & b'_{1k} \\ \vdots & \ddots & \\ b'_{k1} & \ldots & b'_{kk} \end{bmatrix}$.

To ease the readers about our key usage technique, in this subsection we provide an illustrative example. For the sake of simplicity, let's say we are using 8B shuffling and the image is of dimension $16 \times 16 \times 1$ units. So the number of blocks to be shuffled in this case is $\frac{16}{8} \times \frac{16}{8} = 4$. Consequently, the key length is also 4 which consists of unique integers from 1 to 4. Let's assume, `<sender, receiver>` pair agrees on a key $\{4, 1, 2, 3\}$ for their transmission. At the sender side, the secret image is shuffled according to the key, and at the receiver end, the same key is used to recover the actual secret image. For each shuffling process, first we serialize all the blocks, and then use the key as a mapping to rearrange the blocks. At the sender side, equation 2 represents that for each K_i, K_i-th block should be moved to position i, and similarly at the receiver end, i-th block should be moved to position K_i. Fig. 3 visualizes our explanation about the shuffling process.
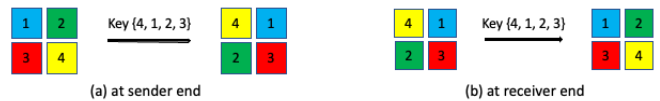

Fig. 3. Shuffling process illustration

V. EXPERIMENT RESULTS AND ANALYSIS

In this section, we first exhibit the security vulnerabilities in vanilla deep steganography model by conducting real attacks that expose confidential data to the adversary. We also present the experimental results of our different keyed approaches,

comparing their performance in terms of secret image reconstruction error, difference between carrier and cover images, computation latency, and human perceptibility. Additionally, we showcase the robustness of our proposed block shuffling approach against surrogate model attacks, cover image availability, steganalysis, and key brute-force attacks.

A. Attack Demos

Our experiment demonstrates that the deep steganographic model alone is insufficient to guarantee adequate security and requires encryption as a backup. To address the threat model defined in Section III-A, we randomly divide our entire dataset into two equal portions, ensuring zero overlaps between the training datasets of the vanilla model and the adversary model. The white-box adversary possesses full knowledge of the vanilla model but differs from it due to disparities in the training dataset. The grey-box adversary has partial knowledge of the vanilla model, specifically the inclusion of prep net, reveal net, and hiding net components. However, they lack information about the filter numbers, kernel size, and layer structures employed in the vanilla model. Consequently, the grey-box adversary replicates the popular image classification model architecture, InceptionV3 [29], and utilizes it as the prep, reveal, and hiding net. The black-box adversary is the least informed, lacking any knowledge of the three components of the vanilla model. Therefore, they mimic another well-known image classification model, VGG16 [30], duplicating it to construct a single encoder and decoder. Table I presents the results of these successful attacks, where the decoded secret images generated by the three adversaries are conspicuous enough to reveal the secret information.

TABLE I
ATTACKS IN VANILLA DEEP STEGANOGRAPHY

B. Steganalysis with Deep Steganography

As defined in II-A, 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 [5] demonstrates the robustness of the deep steganographic method as it succeeds to dodge a traditional lower bit seeking steganographic tool named StegExpose [17]. 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 II 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 II
STEGANALYSIS WITH DEEP STEGANOGRAPHY

C. Performance Analysis

Table III presents quality measures [31] such as RMSE, SSIM, PSNR, and computation latency for different steganography methods. The quality measures are calculated as the average of 10 runs on the test dataset. 4B and 8B shuffling methods outperform vanilla deep steganography in terms of image quality. This is because CNN benefits from additional block creation, as it convolves around images to learn features from blocks or clusters of pixels. On the other hand, 1B and pixel shuffling perform poorly since they scatter pixels indiscriminately, making it difficult for CNN to extract useful features. We also compare our methods with key concatenation, an approach used for text data [11], to demonstrate its applicability to images. While key concatenation performs better in image quality, Table V shows that it performs unsatisfactorily in decryption. Key Concat + 8B shuffling does not offer any advantage over 8B shuffling and actually worsens image quality. We report the encoding-decoding time for the entire test dataset, which is proportional to the depth of shuffling. As the depth of shuffling increases, the number of possible permutations increases, resulting in longer computational time. Pixel shuffling exhibits the highest latency since it shuffles both spatial and channel dimensions.

D. Visual Analysis

Table IV shows the visual representation of secret and cover images before and after encoding along with their amplified differences. In the cases of [5] and [11], 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 [11] 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

TABLE III

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 [5]	8.34	8.59	0.94	0.90	30.23	29.72	**5.91** s	**4.08 s**
Key Concat [11]	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 [11] + 8B Shuffling	8.33	11.34	0.95	0.90	30.00	27.90	6.40 s	4.48 s

TABLE IV

VISUAL ANALYSIS OF DIFFERENT STEGANOGRAPHY METHODS' PERFORMANCES

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 V-C.

TABLE V

EFFECT OF ADVERSARY GENERATED RANDOM KEY

E. Sensitivity to Random Key

As illustrated in Fig. 1, an adversary with access to the decoder (or, a surrogate of the decoder) cannot retrieve the original secret image if we design a proper keyed image steganography. In Table V, we show how the adversary generated random keys perform in revealing the secret image when compared to the correct keys. Note that, Key concat [11] 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.

F. 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 VI, 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 III and V 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.

TABLE VI
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			

G. Availability of Cover Image

The deep image steganography method proposed in [5] 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 VII confirm this limitation of [5] 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.

TABLE VII
EFFECT OF ENCRYPTION

Vanilla Deep Steganography [5]		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

H. Steganalysis

We evaluate the performance of existing and proposed secure deep steganography approaches against steganalysis using *StegExpose* [17], a widely used tool [5], [11]. By employing the default settings of StegExpose, we generate ROC curves, as depicted in Fig. 4. Our experiments demonstrate that the introduction of keyed deep image steganography does not significantly affect the performance of StegExpose compared to vanilla deep steganography [5] or key concatenation [11]. Thus, our proposed model exhibits similar robustness to other image steganography approaches against steganalysis.
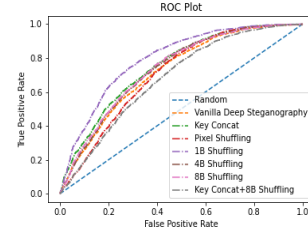

Fig. 4. Performance of existing and proposed approaches against StegExpose

I. Formal Security Analysis

In our system, the input image size is represented as $H \times W \times C$, where H, W, and C correspond to image height, image width, and number of channels, respectively. The number of blocks to be shuffled is denoted by p, and it is defined by $p^2 = \frac{H \times W}{k^2}$ (1). Our SecureImgStego works with both color images and grayscale images, as it is agnostic to the number of channels (C is not present in equation 1).

One can argue that using a secure channel always between a `<sender, receiver>` pair eliminates the need for our system. The key size would increase with larger image sizes, which is a practical concern. However, we defend our design decision by highlighting that key sharing requires only one-time communication through a secure channel. Once the key is shared, the `<sender, receiver>` pair can exchange unlimited secret images of the same size using the same key.

SecureImgStego requires only one shared key for each `<sender, receiver>` pair while enabling the use of multiple NB shuffling models for adaptivity (e.g., $N = 4$ for 4B shuffling). The shared key only needs to be for the model with the smallest block shuffling. For instance, if a pair has a shared key of length 4096 for 1B shuffling, it can also be used for 4B shuffling with an intended key size of 256. The pair can extract integers $1, \ldots, 256$ from the original key without changing their order. Similarly, the pair can derive the key for 8B shuffling (size 64) in a similar manner.

J. Key Space and Security Guarantees

Our system's key represents the correct position of each out-of-position block in the secret image. Thus, the key size is equal to the number of blocks to be shuffled (p^2) [eqn 1]. It is a permutation of unique positive integers from 1 to k^2, and formally: $K = \{K_1, K_2, K_3, \ldots, K_{k^2}\}$ (2) where $K_i \in [1, k^2]$ and each K_i is unique.

Since the key is shared through a secure channel, the only viable approach for an adversary is a brute-force attack. As the key is a permutation of length $k^2!$, the adversary would need to try $k^2!$ keys to uncover the actual secret image. This assumes the adversary has access to an Oracle that can determine the correctness of the constructed secret image. For non-square images, we need to carefully choose k to adapt it in our proposed SecureImgStego. The maximum value k can have is $\gcd(H, W)$. So, formally, $x = \gcd(H, W)$, and $k | x$

For 8B shuffling, the key size is 64, which corresponds to the number of blocks to be shuffled. If the adversary attempts a

brute-force attack to discover the shared secret key, they would need to test approximately $\sim 1.3 \times 10^{89}$ combinations. Even with computing resources capable of testing 10^{17} keys/sec, it would take the adversary 4×10^{64} years to learn the secret key. In the case of coarser-grained shuffling, such as 16B shuffling, key size is smaller and the adversary would require less time to recover the key. In finer-grained shuffling, like 1B shuffling, where the depth of shuffling is higher and the block length is lower, the key size is 4096, significantly increasing the number of key combinations and the time required for the adversary to learn the key. In summary, the security of our model's key is proportional to the depth of block shuffling and inversely proportional to the block length.

VI. CONCLUSION AND FUTURE WORK

In this paper, we expose the inherent vulnerabilities in deep steganography through real attack demonstrations. We propose a keyed image steganography model based on deep convolutional neural networks, where the encryption layer incorporates a secret key to enhance network security. Our model ensures secure communication between sender and receiver, even in the presence of a surrogate model. We demonstrate the effectiveness of our provably secure steganography system in hiding full-size color images while preserving visual characteristics and maintaining confidentiality, even when the carrier image is accessible to the attacker. Our findings reveal the tradeoff between performance and security in different levels of block shuffling granularity. Future work includes extending our approach to asymmetric key-based SecureImg-Stego, exploring applications in audio and video data, and investigating the hiding of multiple secret images within a single cover image using SecureImgStego.

REFERENCES

[1] G. C. Kessler, "An overview of steganography for the computer forensics examiner," *Forensic science communications*, vol. 6, no. 3, pp. 1–27, 2004.

[2] G. C. Kessler and C. Hosmer, "An overview of steganography," in *Advances in Computers*. Elsevier, 2011, vol. 83, no. 2, pp. 51–107.

[3] B. Li, J. He, J. Huang, and Y. Q. Shi, "A survey on image steganography and steganalysis," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 2, no. 2, pp. 142–172, 2011.

[4] M. S. Subhedar and V. H. Mankar, "Current status and key issues in image steganography: A survey," *Computer science review*, vol. 13, pp. 95–113, 2014.

[5] S. Baluja, "Hiding images in plain sight: Deep steganography," in *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, Dec 4-Dec 9 2017, pp. 2069–2079.

[6] P. Wu, Y. Yang, and X. Li, "Stegnet: Mega image steganography capacity with deep convolutional network," *Future Internet*, vol. 10, no. 6, p. 54, 2018.

[7] X. Duan, K. Jia, B. Li, D. Guo, E. Zhang, and C. Qin, "Reversible image steganography scheme based on a u-net structure," *IEEE Access*, vol. 7, no. 6, pp. 9314–9323, 2019.

[8] N. Khan, R. Haan, G. Boktor, M. McComas, and R. Daneshi, "Steganography gan: Cracking steganography with cycle generative adversarial networks," *arXiv preprint*, vol. arXiv:2006, no. 8, 2020.

[9] J. Fridrich and M. Goljan, "Practical steganalysis of digital images: state of the art," in *Security and Watermarking of Multimedia Contents IV*, vol. 4675, no. 1. International Society for Optics and Photonics, 2002, pp. 1–13.

[10] H. Ozer, I. Avcibas, B. Sankur, and N. D. Memon, "Steganalysis of audio based on audio quality metrics," in *Security and Watermarking of Multimedia Contents V*, vol. 5020, no. 2. International Society for Optics and Photonics, 2003, pp. 55–66.

[11] Z. Li, G. Han, S. Guo, and C. Hu, "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)*, 2019, pp. 1937–1944.

[12] K. Sharma, A. Aggarwal, T. Singhania, D. Gupta, and A. Khanna, "Hiding data in images using cryptography and deep neural network," *arXiv preprint arXiv:1912.10413*, 2019.

[13] S. Bharti, S. Behal, and V. Sharma, "Security enhancements for high quality image transaction with hybrid image steganography algorithm," in *2018 Second International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, 2018, pp. 162–169.

[14] X. Duan, D. Guo, N. Liu, B. Li, M. Gou, and C. Qin, "A new high capacity image steganography method combined with image elliptic curve cryptography and deep neural network," *IEEE Access*, vol. 8, pp. 25 777–25 788, 2020.

[15] S. E. El-Khamy, N. O. Korany, and A. G. Mohamed, "A new fuzzy-dna image encryption and steganography technique," *IEEE Access*, vol. 8, pp. 148 935–148 951, 2020.

[16] "Tiny imagenet," http://cs231n.stanford.edu/tiny-imagenet-200.zip, accessed: 2022-09-01.

[17] B. Boehm, "Stegexpose-a tool for detecting lsb steganography," *arXiv preprint arXiv:1410.6656*, 2014.

[18] D. Neeta, K. Snehal, and D. Jacobs, "Implementation of lsb steganography and its evaluation for various bits," in *2006 1st International Conference on Digital Information Management*. IEEE, 2006, pp. 173–178.

[19] E. Wengrowski and K. Dana, "Light field messaging with deep photographic steganography," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, Jun 16-jun 20 2019, pp. 1515–1524.

[20] M. Tancik, B. Mildenhall, and R. Ng, "Stegastamp: Invisible hyperlinks in physical photographs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, CVPRVirtual, Jun 16-jun 18 2020, pp. 2117–2126.

[21] F. Kreuk, Y. Adi, B. Raj, R. Singh, and J. Keshet, "Hide and speak: Deep neural networks for speech steganography," *arXiv preprint*, vol. arXiv:1902, no. 83, 2019.

[22] D. K. Sarmah and A. J. Kulkarni, "Improved cohort intelligence—a high capacity, swift and secure approach on jpeg image steganography," *Journal of information security and applications*, vol. 45, no. 3, pp. 90–106, 2019.

[23] Y. Ding, G. Wu, D. Chen, N. Zhang, L. Gong, M. Cao, and Z. Qin, "Deepedn: A deep-learning-based image encryption and decryption network for internet of medical things," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1504–1518, 2021.

[24] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*. PMLR, 2016, pp. 201–210.

[25] F. Hu, J. Wang, X. Xu, C. Pu, and T. Peng, "Batch image encryption using generated deep features based on stacked autoencoder network," *Mathematical Problems in Engineering*, vol. 2017, 2017.

[26] X. Duan, D. Guo, N. Liu, B. Li, M. Gou, and C. Qin, "A new high capacity image steganography method combined with image elliptic curve cryptography and deep neural network," *IEEE Access*, vol. 8, pp. 25 777–25 788, 2020.

[27] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy (EuroS P)*, 2018, pp. 399–414.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015. [Online]. Available: https://arxiv.org/abs/1512.00567

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[31] U. Sara, M. Akter, and M. S. Uddin, "Image quality assessment through fsim, ssim, mse and psnr—a comparative study," *Journal of Computer and Communications*, vol. 7, no. 3, pp. 8–18, 2019.